The Small Computer Magazine

# kilobaud

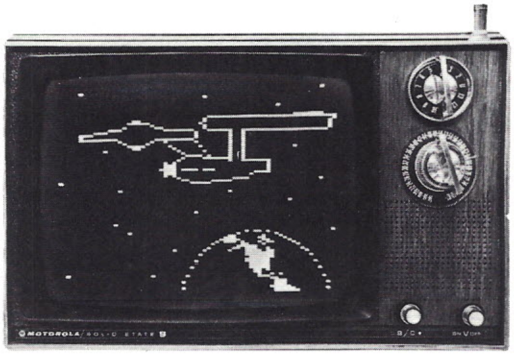
T.M.

**Understandable for beginners . . . interesting for experts**

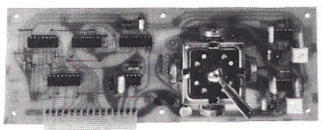
*April 1978 / Issue #16 / $2.00 / DM 7,50 / Sfr 8,10 / Ffr 16,0 / UK £2*

# 6800 COM

## THE COMPLETE COMPUTER SYSTEM FOR ALL APPLICATIONS



GT-61 Graphic Terminal Kit (less monitor). . . **$98.50**

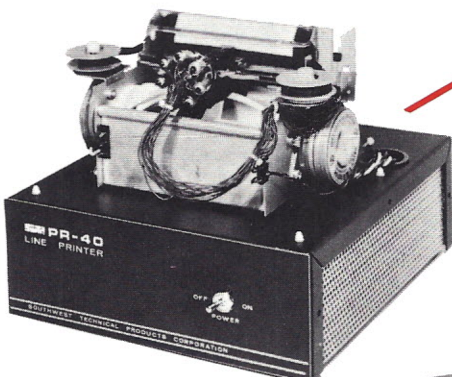


PPG-J Potentiometer Digitizer Kit. . . .**$39.95**

6800 COMPUTER SYSTEM


MP-6800 Kit. . . .**$395.00** MP-6800/2 Kit. . . .**$439.0**

MP-6800/2 (assembled & tested). . . .**$495.00**




PR-40 Alphanumeric Printer Kit
**$250.00**

The SwTPC 6800 computer and its matching peripherals offer you one of the most flexible, reliable and complete systems anywhere. The wide range of completely compatible peripherals makes it possible to tailor the basic computer for all applications; thus, giving you the ease of use and reliability that the 6800 is famous for in all of your computer applications. Why settle for less?

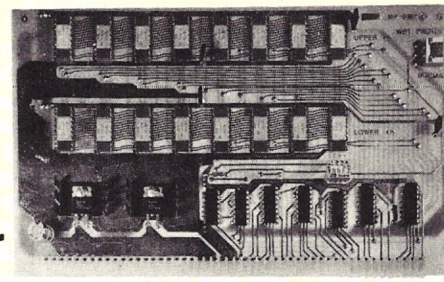

CT-64 Terminal System Kit (less monitor)
**$325.00**

CT-VM Video Monitor for CT-64
**$175.00**

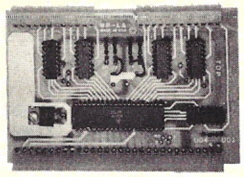
MAIL
COUPON
TODAY

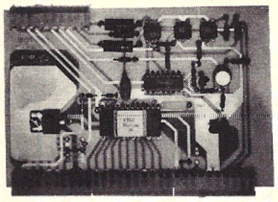
# PUTER SYSTEM

MP-8M Memory Board Kit. . . .**$250.00**
MP-16 Memory Board (assembled & tested). . . .**$400.00**

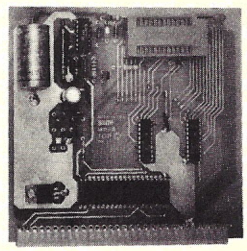


MP-LA Parallel Interface Board Kit
**$35.00**

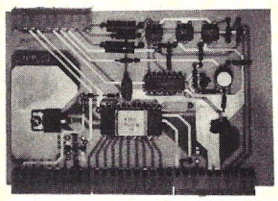




MP-S Serial Interface Kit
**$35.00**



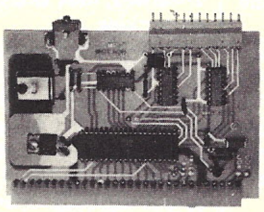
MP-R 2716 EPROM Programmer Kit
**$44.95**



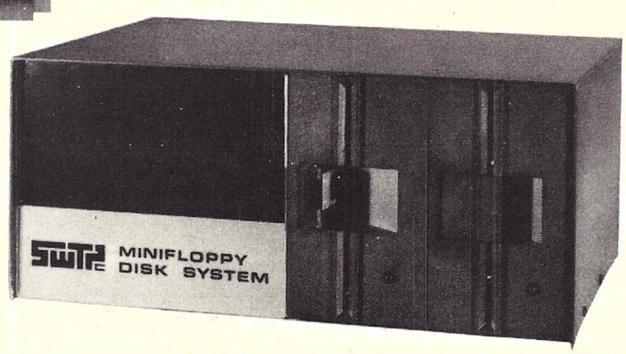
MP-C Control Interface Kit
(Serial). . . .**$40.00**


AC-30 Audio Cassette
Interface Kit. . . .**$79.50**

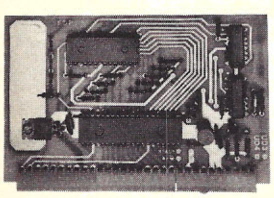
MP-T Interrupt Timer Interface Kit
**$39.95**



MF-68 Floppy
Disk Kit. . . .**$995.00**




6800 SOFTWARE
**$4.95 & 9.95**

MP-N Calculator Interface Kit
**$46.50**

---

**Please send item(s) checked. (All prices ppd. Continental U.S.)**

- — MP-6800
- — MP-6800/2
- — GT-61
- — PPG-J
- — PR-40
- — CT-64
- — CT-VM
- — MP-S
- — MP-LA
- — MP-8M
- — MP-16
- — MP-R
- — MP-C
- — AC-30
- — MF-68
- — MP-T
- — MP-N

☐ Enclosed is $________ or BAC #________________
or Master Charge #________ Bank #______ Expire Date________
Name________________________________
Address________________________________
City________________ State ________ Zip________
☐ Please send catalogue of complete 6800 Computer System products.

SOUTHWEST TECHNICAL PRODUCTS CORPORATION, Box 32040, San Antonio, Texas 78284

# PUBLISHER'S REMARKS

*Wayne Green*

### Go off the Deep End

Letters from readers and questions posed during forums at microcomputer shows point up the *Catch-22* nature of not really understanding microcomputers. A recent reader poll indicated that roughly 25 percent of *Kilobaud* readers are holding back from buying their first microcomputer system. The problem is relatively simple: just reading articles about microcomputers isn't enough to impart a real understanding; you have to have one in hand to use along with the articles if you are going to make much progress. That's fine, but without understanding, how can a person make an intelligent choice of microcomputer(s)?

So, we have tens of thousands of people who are desperately trying to read enough to understand what they should buy . . . but who are unable to understand because they haven't bought.

The way out of this mental house of mirrors is easy: Flip a coin and buy any of the low-cost familiarizer systems—the KIM, the Elf, the MEK, the Heath ET-3400, the E&L, etc. A lot of low-cost microcomputers are available, any one of which will get you going. In fact, they will probably be far more valuable to you than one of the larger systems because their simplicity forces you to really learn how they work—both hardware and software—and this is your main goal. Buyers of larger systems are tending to try to go the black-box route, which means they want to shortcut their learning cycle by substituting hardware they don't understand, but which is reliable. Great—you can play games right away, but then you don't really understand what is going on when you want to start adding things to your system.

Most of these low-cost systems can be expanded almost beyond belief. Just look at what is happening with the KIM! Entire books on expanding the KIM system are coming out, and MOS Tech has a SuperKIM being readied.

Little that you might eventually want out of microcomputers won't be enhanced by working with one of these small systems for starters. Their start-up price is low enough to fit just about any budget (some are under $200); but for the most part, plan on about $250 for your first setup. Dealers tell me that the used value of these systems drops very little, so you could hardly ask for a lower-cost education. You'll be able to buy a system, use it and learn, and then get a good part of your cash investment back as you progress to a bigger system . . . if you are ever able to part with your first love.

These low-cost systems are based on the same chips being used by full-blown microcomputers, so you can, if you want, stick with your mini-micro and just add to it . . . memory, S-100 bus, floppies, printer, etc.

The main message is this: Stop making excuses about not being able to decide or being short of money; get a small developmental or training system and get started with the most important parts—having fun and learning. Every day you wait you are missing out on excitement and education . . . you are paying much too dearly for your procrastination.

### Waiting for Better Prices?

A recent reader poll indicated that there are still thousands of potential computer users who are hanging around waiting for a drop in prices comparable to calculator and digital-watch price drops. It isn't going to happen.

Oh, we'll have some gradually lowering prices, but no catastrophic price reductions are in prospect in the foreseeable future.

Memory will be coming down in price on a fairly steady curve as bigger chips are made and mass-production techniques reduce costs. With 8080A chips now below $10, how much more can you save on a CPU? Bringing the 8080A down to $5 or even adding some memory to it won't cut things much.

Once we get some business systems into production we will begin to see price reductions. A 12 percent cost reduction is assumed when production is doubled, so a good, large run of computers could bring cost benefits. This is still a way down the line. I haven't made it a secret that we are laying the groundwork for a microcomputer publication aimed at small business. I haven't rushed to get the first issue out because the fundamental message would have to be: not much for you yet . . . perhaps next year.

If you've been watching the prices for used computer systems, you've noticed that they are staying high; thus, you don't lose a lot if you buy a system, use it for a while and then sell it. My advice is to buy that computer now and have your fun—don't sit around waiting for prices to drop and find that you've missed out.

What should you buy? Start with any system that appeals to you and then go to the next . . . and the next. The more systems you work with and understand, the more fun you'll have and the more you'll be worth.

Money is probably a problem; so perhaps you should start with one of the mini-micro systems such as the KIM. You want to learn and have fun, and any of these will provide plenty of opportunities. The next step up might be to a PET, TRS-80 or H8. I'm getting rave letters from readers on all of the above systems.

The people who sat around waiting for television set prices to go down missed several years of great entertainment. Don't miss the fun . . . get a computer and join in. You'll have a wonderful time.

### Why Equipment Doesn't Work

At one of the NCC sessions, a speaker from one of the top microcomputer manufacturers explained why so many hobbyists have had trouble getting their systems working. He pointed out that it costs a lot of money to get every bug out of a piece of equipment and that one solution to this problem is to start shipping the hardware at some early point in the design cycle and let the users finish the design. In this way, you have the money coming in from the sales and hundreds of technicians working on your board. You do your engineering by opening the mail. The speaker said that all of the major firms have done this, although most of the larger firms don't do it any more.

That goes a long way toward explaining why the first system I got, even though it was factory assembled, took almost a year to get working.

A lot of this nonsense would be eliminated if hobbyists and dealers would take the trouble to put their gripes in writing and send them to the manufacturer, with a copy to me. I get pretty upset when I talk with someone at a computer show and hear some terrible story of his being victimized by a manufacturer . . . and never doing anything about it. With one exception, we've had considerable success in getting manufacturers to clean up their acts.

### *Reader Responsibility*

One of your responsibilities, as a reader of *Kilobaud*, is to aid and abet the increasing of circulation and advertising, both of which will bring you the same benefit: a larger and even better magazine. You can help by encouraging your friends to subscribe to *Kilobaud*. Remember that subscriptions are guaranteed—money back if not delighted, so no one can lose. You can also help by tearing out one of the cards just inside the back cover and circling the replies you'd like to see: catalogues, spec sheets, etc. Advertisers put a lot of trust in these reader requests for information. To make it even more worth your while to send in the card, a drawing will be held each month and the winner will get a lifetime subscription to *Kilobaud*!

### A Call for Papers

Something odd seems to happen to hobbyists when a computer show issues a call for papers. Paper-writers spring up everywhere, ready to donate their hard-earned knowledge to just about anyone who asks.

In many cases, the same amount of work would result in an article that could be published

*(continued on page 20)*

# EDITOR'S REMARKS

*John Craig*

### Look Out Sears!

Have you taken a look at the inside cover of your new Montgomery Ward spring & summer catalogue? The Cybervision™ *home computer* has arrived! All of us in the personal-computing field have been expecting this for some time . . . it was just a question of who was going to be first. Montgomery Ward definitely has the jump on the others, and I like their approach. They're not afraid to call it a home-computer system . . . and they devoted two full pages to the ad. (I bet there was some discussion on whether they should call it a "computer system" for fear of scaring off individuals who have preconceived notions about computers and how they do more harm than good.)

The unit is designed to work with a black and white or color TV and, therefore, doesn't come with a monitor. A cassette recorder is mounted in the top, with slots for storing 12 cassettes. There are two calculator-type keyboards provided with the unit (full alphabet and digits 0-9).

As you may have guessed, the hardware isn't all that impressive, and, for a computer hobbyist, the ad leaves a lot more unanswered questions than answered ones. Not that it's of any great importance, but it would be interesting to find out what company is behind the Cybervision. On the *important* side: What kind of microprocessor is the system built around . . . what is the memory size . . . are there plans for future expansion of the memory . . . BASIC . . . assembly language . . . how about an ASCII keyboard interface . . . floppy disks . . . and, most important, what kind of printer is in the works? Actually, the real question might be, "Are *any* of those items in the works?"

The people at Montgomery Ward have enough faith in this product to give it prime "billing" in their catalogue. Without a doubt, that faith is not based on the hardware I've just discussed. No, it's the software that's going to make or break any personal computer . . . and the Cybervision has an impressive array (including some I never thought of!). The games are there (of course) and it looks as if most of the popular video games are available, or coming up in the future. (One of the things that bothered me about the list of upcoming programs is that there were specific dates when it would be available. It had better already be developed or someone is kidding someone else about those delivery dates . . . but they can't kid us.)

Along with the Game Series, MW is offering a Home Series (which contains such programs as income-tax preparation, calculator, vegetable gardening, etc.). I guess I wasn't prepared for nursery stories on the home computer just yet . . . but MW has a bunch of 'em in their Story Series. You may not be ready for Hansel and Gretel on your home system, but the kids should love it!

I was tickled pink to see that the main emphasis in software is in the Educational Series (16 programs scheduled, as opposed to ten for Game, nine for the Story and five for the Home series, respectively). I hope the programs are as good as they sound; if they are they'll succeed in getting a lot of people turned on to home computers. I feel the most benefit to be derived from home computers in the years to come will be in the educational area.

I'll have to get in touch with our vast underground network to see if I can't get the answers to some of those questions posed earlier.

### Kilobaud Klassroom

You've no doubt noticed that Kilobaud Klassroom has been absent from the pages of *Kilobaud* for two consecutive months. Some unfortunate incidents beyond our control were responsible . . . and we're as sorry about it as all of you who are following the series. George will be back next month . . . bear with us.

### Heard Any Good Stories Lately?

Humor always seems to be in short supply in technical/hobby publications; it shouldn't be that way. If you have any humorous incidents, short stories or anecdotes you'd like to share with the rest of us, then drop me a line. (Cartoon ideas are fine, too.)

### User Groups and New Newsletters

*CP/M Users' Group.* Hey, this is going to be a biggie! Tony Gold and a few associates probably have one of the first successful software exchange networks in the country going . . . and going strong! What's really great is that the service is, for all practical purposes, free; and the software Tony et al provide is all in the public domain. And, what software! At the time this is being written they have 14 volumes available. Fourteen volumes = fourteen diskettes! That's more software than you'd probably ever be able to use.

Each volume is available for $8, which covers the cost of the diskette and a small copying fee. (Tony's mailbox does not handle blank diskettes . . . so don't send 'em.) They're also offering Microsoft BASIC and Microsoft FORTRAN through the users' group . . . at just a few dollars above dealer cost. For you North Star owners, CP/M is also available (as a commercial product) on minidiskettes through the group.

Drop Tony a line and ask him to send you a copy of the incredible list of software CP/M Users' Group has available (CP/M Users' Group Notes). The cost is $4 for joining the group. (As a side note, they're encouraging the formation of local user groups across the country.)

Tony Gold, CP/M Users' Group, 345 East 86 Street, New York NY 10028.

*1802 Owners. Ipso Facto* is a publication of the Association of Computer Experimenters and has some good down-to-earth stuff for you 1802 home-brewers. Some of the material I looked over contained articles on interfacing Don Lancaster's TVT-6, building a hex display, cassette interfaces and more.

Tom Crawford, 50 Brentwood Dr., Stoney Creek Ontario Canada L8G 2W8.

*International Computer Center Director*. Well, you 6800

*(continued on page 20)*

**kilobaud**™

**Publisher**
Wayne Green
**Executive Vice President**
Sherry Smythe
**Editor**
John Craig
**Managing Editor**
John Barry
**Editorial Assistants**
Dennis Brisson
Susan Gross
**Production Department**
Lynn Panciera-Fraser
Craig Brown
Gayle Cabana
Robert Drew
Michael Murphy
Weston Parker
Bob Sawyer
Noel R. Self
Robin M. Sloan
**Typesetting**
Barbara J. Latti
Pauline Halvonik
Jennifer Johansson
Marie Walz
**Photography**
Bill Heydolph
Tedd Cluff
**Associate Editors**
Don Alexander
Tim Barry
Sheila Clarke
Rich Didday
Phil Feldman
Tom Rugg
Peter Stark
**Bookkeeper**
Knud E. M. Keller
**Assistant Bookkeepers**
Nancy Dupuis
Frances Marion
Dominique Smith
Cida Teixeira
**Marketing**
Sherry Smythe
Jack Shimek
Cynthia Gray
**Circulation**
Dorothy Gibson
Rhonda Ramsey
Florence Goldman
Valerie Horn
**Receptionist**
Doni-Anne Jarvis
**Computer Data Control**
Judy Waterman
Judy Brumaghim
Linda Cate
Mary Kinzel
**Computer Programming**
Ron Cooke
Richard Dykema
Steven Lionel
**Printing**
Michael Potter
William Cering
Dwight Perry
**Mailroom**
Theresa Toussaint
Sue Chandler
Ethan Perry
Bill Barry
Stephen Coombs
Noreen Parker
**Plant Maintenance**
Lorraine Pickering
Mike Coutu
Mike Baird
**Advertising**
Heidi Kulish
Marcia Stone
Ila K. Witty
**European Distributor**
Monika Nedela
**Australian Distributor**
Katherine Thirkell

Kilobaud is published monthly by 1001001, Inc., Peterborough NH 03458. Subscription rates in the U.S. and Canada are $15 for one year and $36 for three years. In Europe: Kilobaud erscheint montatlich bei Fachzeitschriftenvertrieb Monika Nedela, 7778 Markdorf, Markstr. 3. Abonnement DM 70.- plus Porto 7.20 + gesetzl. MWST; Sfr. 81.- plus Porto 7.20. Australia: For subscriptions write—Katherine Thirkell, Sontron Instruments, 17 Arawatta St., Carnegie, Vic. 3163 Australia. Please write for other foreign rates. Second-class postage paid at Peterborough NH 03458 and at additional mailing offices. Publication No. 346690. Phone: 603-924-3873.  INCLUDE OLD ADDRESS AND ZIP CODE WITH ADDRESS CHANGE NOTIFICATION.

# AROUND THE INDUSTRY

*John Craig*

### Computer Retailers' Association Formed

The Computer Retailers' Association, a trade association of computer stores, has been formed with 24 founding members, including stores from across the United States, and one Canadian store. The objective of the association is to provide services that individual stores cannot effectively provide themselves. Examples of such services include compiling industry statistics, working with manufacturers to improve the relationship between computer stores and manufacturers, arranging for group insurance, providing information about the computer-store business to the financial community and encouraging high standards among computer retailers. Specific objectives will be determined at a later date by the association's membership.

### How Did the Association Get Started?

The First West Coast Computer Faire in April 1977 devoted a session to computer retailing. Just before the session there was an informal meeting of several computer-store owners. There was strong agreement that an association of computer stores was needed. Portia Isaacson suggested that a meeting for computer-store owners be held at the National Computer Conference in June 1977 to get the planning for the association underway.

Computer-store owners were invited to the NCC meeting by direct mail, through magazine announcements and by telephone solicitation. Prior to the NCC, about 50 stores were polled by telephone to determine their level of interest in an association of computer stores. The response was overwhelmingly positive.

The NCC meeting was chaired by Ray Borrill and attended by 30 to 40 computer-store owners. Again, there was universal agreement on the general need for an association.

Two significant events took place at the meeting. First, Los Angeles attorney Kenneth S. Widelitz, author of *Kilobaud*'s Legal/Business Forum, presented a proposal for a Computer Retailers' Association and a specific plan for forming it. Second, the Computer Retailers' Association Committee composed of computer-store owners chaired by Portia Isaacson was formed to implement and support Mr. Widelitz's plan. The essential steps in the plan to form the Computer Retailers' Association were:

1. Mr. Widelitz would establish an interest-bearing Computer Retailers' Association Trust Account.

2. Dr. Isaacson would mail a letter to all known computer stores and prepare a news release explaining the plan for forming the association and asking that computer stores indicate their interest by sending a $100 check to the Computer Retailers' Association Trust Account.

3. The association would be incorporated in California by Mr. Widelitz if 20 stores had responded by November 15, 1977. If 20 stores failed to indicate interest, all money in the trust account would be returned.

Subsequent to the NCC, the chairman of the Computer Retailers' Association Committee did mail a letter to about 250 computer stores asking their support for the proposed association. A news release to all computer-industry publications was also mailed. A questionnaire was included with the letter in order to determine the level of interest and get opinions on the possible activities of the association. The results of that survey can be obtained from the Micro Store, 634 S. Central Expy., Richardson TX 75080.

In August, at Personal Computing '77 in Atlantic City, two meetings of computer-store owners were chaired by Portia Isaacson. The meetings were well attended, not only by about 50 computer-store owners, but also by other interested industry people. In December, the Computer Retailers' Association was incorporated as a trade association under the laws of the state of California.

# LEGAL/BUSINESS FORUM

*Kenneth S. Widelitz*
*Attorney-at-Law*

In the November Legal/Business Forum I discussed some of the philosophical issues raised in arguments surrounding the question of whether or not computer software should be protected by copyright. Although the software subcommittee of the National Commission on New Technological Uses of Copyrighting Works (CONTU) has not yet issued its final report, its preliminary report did indicate that it believed computer programs should be protected by copyright.

In November, I specifically avoided getting into any of the details of the new Copyright Act passed by Congress in 1976, effective January 1, 1978. A letter from Verlynn J. Johnson in the October *Kilobaud* raised some questions about copyright which John Craig, in his editorial reply, indicated would be covered in the Legal/Business Forum. OK, John, I can take a hint.

In his letter, Verlynn indicates that he is getting into systems programming but has no intention of coming up with an operating system completely from scratch. He wants to know if he can incorporate previously published routines and subroutines into his operating system and, if so, if he can copyright the operating system. In order to get to the issues involved in answering this question, it is necessary to understand the basic workings of copyright.

### To Promote the Progress of Science

The Copyright Act was enacted by Congress pursuant to its power under the Constitution, Article I, Section 8, Clause 8, which grants Congress the power "to promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries."

The underlying rationale for the Copyright Law provides incentives for the creation and distribution of original works of value to society. The ultimate hurdle the Copyright Law must overcome is how to balance the societal interests of the broad dissemination of creative works with the interests of individual authors in protecting their property in order that they may financially benefit from that which they have created.

### Caveat

Section 117 of the new Copyright Law specifically states that the new law does not give the owner of copyrighted software any greater or lesser rights than those afforded under the old law. However, Section 117 only relates to that bundle of rights I have discussed under "What Are Your Copyright Rights," which follows.

I have discussed those rights in terms of the new law because I don't think the final CONTU report will significantly change them. I say that because CONTU's proposed replacement for the new Section 117 doesn't appear to substantially change any copyright rights.

The new law's provisions relating to term of protection, formal requirements, infringement, remedies and other concepts discussed in this Forum do not come within the scope of the existing new Section 117 and are applicable to computer programs.

### Nuts and Bolts of Copyright

The new Copyright Law fixes the duration of copyright for works created on or after January 1, 1978, for a term consisting of the life of the author plus 50 years. Copyright protection of a work begins at the work's creation under the new law.

Under the old law there were two forms of copyright: common law and statutory; the common-law scheme protected works until they were published, at which time the statutory scheme set in. This made publication an important occurrence, and much of the courts' time was spent in defining what it meant.

Under the new law publication is no longer as great a concern to authors. However, it still does play *some* role in copyright. The duration of copyright for a work made for hire is 75 years from the year of its first publication or 100 years from the year of its creation, whichever expires first.

### Definitions

Some definitions are now in order. A work is *created* when it is fixed in a copy for the first time. *Copies* are material objects in which a work is fixed in any method now known or later developed, and from which the work can be perceived, reproduced or otherwise communicated, either directly or with the aid of a machine or device. *Publication* is the distribution of copies of a work to the public by sale, lease or lending. A *work made for hire* is a work prepared by an employee within the scope of his or her employment, or a work specially ordered or commissioned for use as a contribution to a collective or supplementary work, among other conditions, if the parties expressly agree in a written instrument signed by them that the work shall be considered a work made for hire.

Let's sort some of this stuff out. You are sitting at your TVT writing a program to predict the date on which Tralfamadorians will next communicate with Earth. As you type in the listing you are creating a copy. Your copyright exists in your program from the instant you press a key. However, if you are pressing the keys of your employer's computer, you are creating a work made for hire and do not own any copyright. If you are working as an independent contractor and are creating software for someone else's use, you are in a gray area. In such a situation, you should have a written agreement spelling out who will own the copyright.

### What Are Your Copyright Rights?

If you are the owner of a copyright, you have the right to do or authorize any of the following: (1) reproduce the copyrighted work in copies; (2) prepare derivative works based upon the copyrighted work; (3) distribute copies of the copyrighted work to the public by sale or other transfer of ownership, or by rental, lease or lending. Each of these rights is an independent right; that is, you can enter into a contract with a software house that may make and sell copies of your program, but you can retain the right to prepare derivative works. You may license a printer to make copies of your listing and reserve the right to sell such copies for yourself.

However, once you sell an authorized copy of your program, you lose control over that authorized copy. That is, if I buy a copy from you, I then own the copy. I can keep and use the copy, or I can sell it or lease it. However, I cannot make another copy of the copy and then sell or lease the original while retaining the copy that I have made for my own use. That would be an infringement of the copyright. More on that later.

One of the rights previously mentioned is the right to prepare derivative works based upon the copyrighted work. The problems raised by this right directly relate to the answer to Verlynn's question. The definition of a *derivative work* appears in Section 101 (quoted in full below) of the Copyright Act.

*A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship is a "derivative work."*

Nothing in that definition touches directly on the use of routines or subroutines. In fact, much of it relates to creative endeavors not at all related to computer programming. I hope the following discussion will tie the foregoing definition into an answer to Verlynn's question.

Professor Nimmer, whose treatise on copyright is cited in virtually every case dealing with the subject, has the following to say about derivative works.

*If that which is borrowed consists merely of ideas and not of the expression of ideas, then although the work may have in part been derived from prior works, it is not a derived work. Put in another way, a work will be considered a derivative work only if it would be considered an infringing work if the material which it has derived from a prior work had been taken without the consent of a copyright proprietor of such a prior work.*

To comprehend what Nimmer is saying we must first understand that in no case does copyright protection extend to any idea, procedure, process, system, method of operation or concept. That is, copyright protection extends only to the *expression* of ideas, not to the ideas themselves. However, some ideas, concepts, procedures or processes are so basic and fundamental that copyright does not protect an explanation of them. For instance, consider your BASIC bubble sort technique.

It is certainly a fundamental procedure or process. It may appear in a copyrighted program. You may look at that bubble sort in that copyrighted program and say, "Oh boy, that's just what I need." You may take that bubble sort routine without getting the permission of the copyright owner and not have infringed on the copyright. If you then use that routine in your program, your program is not a derivative work, although your program may have in part been derived from the prior work. Of course, the question is, how far can you go?

You can go only as far as you are taking an idea, procedure, process or concept. You cannot take the expression of such idea, etc., because, as the courts have stated, "The entirety of the copyright is the property of the author; and it is no defense that another person has appropriated a part, and not the whole, of any property."

Therefore, it seems that unless it is a *very* basic building block, you are probably infringing on a copyright. Certainly you are infringing on the copyright if you extract a routine that consists of several of those basic building blocks tied together to accomplish a specific task.

If there is a fairly sophisticated subroutine you wish to incorporate into a program you are writing, you had better get permission from the copyright owner to use that subroutine. It is perfectly proper to use monetary incentives to obtain such permission; I am sure virtually all owners of software copyrights wrote the software with such monetary incentives in mind.

The only time you don't need the copyright owner's permission is when the work is in the *public domain*. Under the old law, a work was placed in the public domain if it was published without the affixation of a copyright notice. Under the new law, if a work is published without a copyright notice, it will still be subject to statutory protection if not more than a relatively small number of copies have been publicly distributed without notice, or if the work is registered within five years of publication without the appropriate notices and a reasonable effort is made to add the appropriate notice to the copies that have been publicly distributed.

However, even if only a relatively small number of copies have been publicly distributed without the notice and if no effort is made to correct that error or if the work is not registered within five years, the work will go into the public domain.

### Notice Requirements

The notice requirements referred to above are quite simple. They consist of a C in a circle, the word copyright or the abbreviation copr.; the year of the first publication; and the name of the copyright owner (i.e., Copyright, 1978, by Kenneth S. Widelitz). The notice must be affixed to copies in such a manner and in such location as to give reasonable notice of the claim of copyright. The Register of Copyright prescribes regulations regarding the exact positioning for various creative works.

### Deposit and Registration

Deposit of copies of copyrighted materials with the Library of Congress and registration of copyrighted materials with the Register of Copyrights are separate, although closely related,

*(continued on page 21)*

# THE BASIC FORUM

John Arnold/Dick Whipple

Here we are again, with a desk piled high with letters from BASIC Forum readers. If this keeps up (and we hope it does!), we will have to use a computer to help keep track of reader responses. We want to say that we do our best to schedule your material into the Forum as soon as possible after receipt, but with so much coming in and such limited space, you can expect a publication delay of three or four months. We also regret that occasionally letters get completely squeezed out of our column. The reasons are mostly technical (name-address missing, etc.), not related to subject matter or point of view. At any rate, folks, keep those cards and letters coming in.

### How Effective is BASIC?

In past Forums we have made comments regarding the effectiveness of BASIC as a language for beginning programmers. We've received several replies, and would like to present some of them at this time. (We will make further personal comments later on.)

The first comes from Richard Williams, 135 Harrison St., Apt. B, Dekalb IL 60115. He writes: "Richard Blumenfeld of Brewster NY touched upon a very important topic when he mentioned the ever-increasing number of instructions being implemented in the BASIC language. The problem (and it is a problem) will be corrected by the implementation of a comprehensive high-level language for those who have advanced beyond the 'primer school' BASIC.

"After years of association with many languages, I would recommend PL/1 as *the* high-level language and leave BASIC to be used as it is intended to be—by the beginner.

"PL/1 already includes the capabilities of FORTRAN, BASIC, COBOL, RPG and several lesser-known languages such as SNOBOL, LISP, etc.; because of this it does not need the massive alterations as does BASIC for it to be fully capable of handling the many needs of the microcomputer user."

Another letter expressing a similar thought, but advocating a different language, comes from Ray Van De Walker, 212 D Nashville St., Hunt [*sic*] Beach CA 92648. "I think that most advanced BASIC users have gotten into a rut because BASIC is *inherently* a limited language. It was *designed* that way to make it less intimidating.

"The first language I learned was APL. A couple years ago, I finally (grudgingly) learned FORTRAN. Two months ago I learned BASIC. Frankly, the whole process of learning these other languages has taught me that it is simply *amazing* what people will put up with when they don't know any better. Until I learned COBOL I didn't really believe that people wrote 10,000-line programs (100 lines of APL can be made nearly omniscient).

"I do *not* wish to disparage or attack BASIC; it is a wonderful language—for simple programs. I really feel, though, that if you're getting bored, or are *really* tired of 600-line BASIC programs, then perhaps it's time you learned APL. I don't know of any microcomputer APLs that are running (manufacturers dislike the additional character set; standard APL requires terminals that overstrike). Somewhere out there (I've heard), there is an association of amateurs trying to roll their own. I'd dearly love to hear from you. Also, *imagine* how much I'd be willing to pay for a working APL (there's quite a group of us around here) on a common micro.

"The major advantages of APL are:

1. Source language is much less bulky.
2. Source language looping is almost never used—most common data processing functions are primitives (and easy to use). Because of this, APL programs tend to run *very* fast. (Sometimes even faster than comparable Assembly programs; the primitives are generally better written machine language.)
3. Conditional branching is user written (*not* a feature built into the language), thus very flexible.
4. Any program in a work space (working file) can use any other program as a subroutine.
5. Local variables, array operations, text execution, execute this program when an error occurs.

"Enough propaganda and perhaps you can begin to see how people can become APL fanatics. Writing games is *faster!* Businesses can use throwaway code for even the most demanding programs (throwaway code means it's easier to write a new program than modify an old one). I'd like to hear from you people out there—write to the Forum or to me."

Finally, we present a letter that is not so much for any other language as it is against BASIC. D. A. Harrod, PO Box 9475, Rochester NY 14604, has this to say: "Regarding your letter from Clive Grant (BASIC Forum, November 1977), I really don't see how he could have learned ALGOL in 1952 since Backus didn't describe it to the international committee until 1960! (I'd also like to know what he ran it on. ENIAC?)

"I pay my rent by programming in an extended FORTRAN that has a Double Complex Hyperbolic Tangent function (64 bits for the real and 64 bits for the imaginary part, 16 bytes in all). On a machine that has 16 registers, the only use I can find for BASIC is to play games like Star Trek. Why? Because BASIC is a language invented to teach people that computers are nothing to be afraid of, and it's a nice term project for systems-software science majors to write BASIC for their assemblers, interpreters and compilers course (maybe only for extra credit).

"The only reason BASIC is so popular is that it's easy to write for a machine that doesn't have registers (and really, an 8080, 6800 or 6502 only has an accumulator), and anyone can teach his 12-year-old how to write programs in an afternoon.

"There's no way around it . . . BASIC is trivial, a kludge on the way to SNOBOL. An interpreted language is by definition *slow* and requires overhead. You will always make out better with a language that incorporates dynamic memory allocation (it puts data wherever there's free space and does "garbage collection" when it runs out of room).

"Disk BASIC is a real mess . . . better to use your disk for a compiler to generate machine code that takes up 20 percent of the room a BASIC program would occupy, and a good relocating loader to support a library of functions that are loaded as needed (why keep the code for SIN, COS and TAN in core if all you need is LOG . . . maybe you don't need any of them for a particular program).

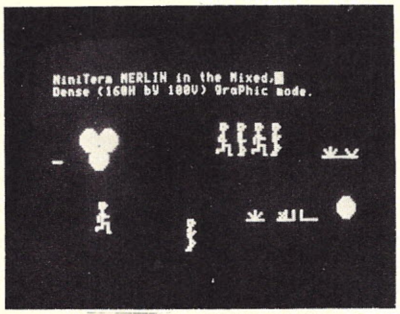
"What really burns me is that no one talks about the language used for writing these BASIC interpreters: PL/M for the 8080, and MPL for the 6800. These are high-level languages written by the manufacturers of these chips strictly for the people who plan to earn a living from computers, and, I guess, therein lies the problem. The difference between hobbyist and professional is that what's "fun" for one is the "bread and butter" for the other; and no one can advance from one to the other until he puts BASIC in its proper perspective . . . it's a three-wheeled velocipede in a world of Harley-Davidsons, a child's toy, to be discarded long before puberty.

"Perhaps I'll step on a lot of toes by saying this, but anyone who spends $300 for memory to run XYZ-SUPER-BASIC (or FOCAL, which is one vendor's version of BASIC) might as well hire a chauffeur to drive his Volkswagen; and I have a bridge I'd love to sell him real cheap (it connects Manhattan to another borough of NYC, and Frank Sinatra sang a song on it in a movie, a long time ago)."

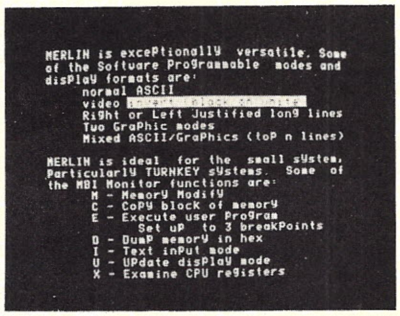
Although we appreciate the position of experienced and professional programmers, many of our readers and a large number of computer hobbyists are, after all, *beginners.* BASIC fills a definite need for them as it serves to introduce them to programming in a rapid and convenient way. The fact is, most hobbyists don't have disk operating systems with "Extended FORTRAN" and "Double Complex Hyperbolic Functions," nor do they have a need for such sophistication. As a beginning programmer develops his skills, he will naturally seek out more elaborate hardware and software capability. But he has to start somewhere, and we see nothing wrong with starting on a small machine with a BASIC interpreter.

A point often ignored by detractors of BASIC is the convenient manner in which programs can be entered and modified. This capability is of great importance to the novice who spends much of his time experimenting with programming techniques. After all, we are not born knowing the tricks of the computer

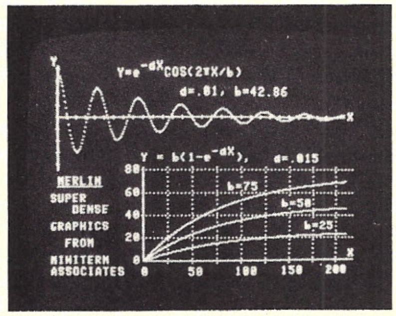
# The many faces of MERLIN



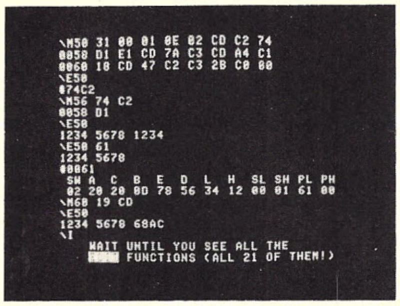

Dense Mode: 160H × 100V
Running Man Patterns



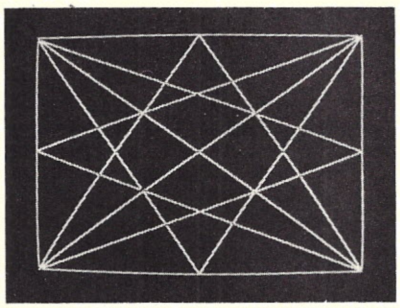

Propaganda



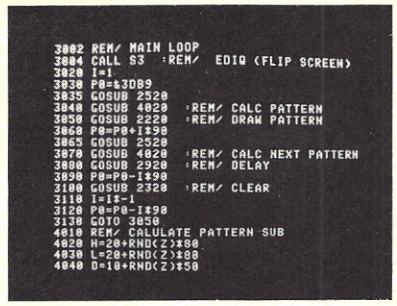

Super Dense: 320H × 200V
Equation Plotting



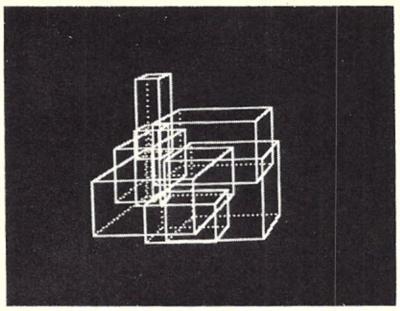

Monitor Debug Usage



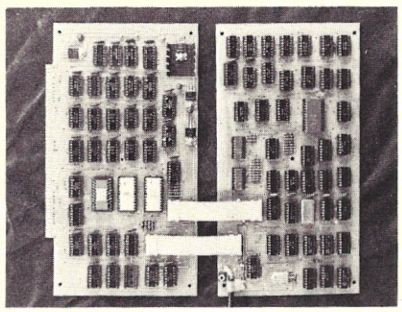
Super Dense: 320H × 200V
Line Drawing

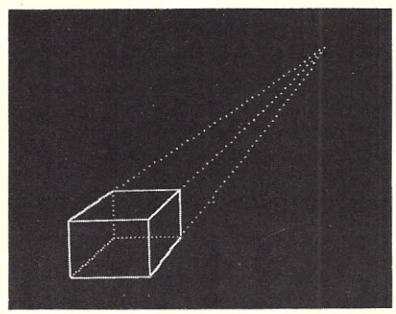



BASIC Program Listing
Output Shown Below



Super Dense: 320H × 200V
3-D Boxes





Super Dense: 320H × 200V
Perspective Drawing

## **MERLIN** (and your S-100 Computer...)
## ... the graphics development package you've been waiting for.

Do these photos suggest an application in your field? Whether you're into architecture, astrology, music or whatever, you can apply graphics to make your presentations more effective and your work more efficient.

**Real Time Plotting**
- Heart Rate
- Navigation
- Spectrum Analysis

**Complex Equation Plotting**
- Stock Market Trends
- Teaching: Mathematics
- Circuit Responses

**Fine Line Drawings**
- Architecture
- 3-D Projections
- Circuit Layouts

**Pattern Movement**
- Animation
- Games

**MERLIN** is also a multipurpose system monitor board with ROM monitor and editor software, parallel keyboard port, and audio cassetts storage, besides being your text and graphics video output device.

Write or call today for your free copy of our new catalog, an index to available application notes, and a list of **MiniTerm** Dealers.

**MERLIN** is supported by expert technical assistance, a constantly expanding series of application notes, and a newly formed users group.

Assembled and tested **MERLINs** start at $349, $269 for kits. Super Dense add-on, firmware, and cassette interface are extra.

**MiniTerm Associates, inc.**

**Dundee Park, Andover, MA 01810 (617) 470-0525** **M40**

trade—they are acquired! Compilers are nice for production work where time and memory efficiency are practical necessities. Compiled programs often require a multistep process (compilation, assembly, loading and execution), which discourages program modification and, hence, experimentation. Although "load and go" compilers exist on big machines, we know of none for micros. The interpretive mode of BASIC, though slower and less memory efficient, actually encourages experimentation and aids the learning process.

## Computing Arc Sin and Arc Cos

In the November Forum, a contributor needed a way to compute arc cos and arc sin. Several readers responded with suggestions that we thought might be of interest. The first comes from W. R. Ayers, 26969 Moody Rd., Los Altos CA 94022. He writes: "Jim Faliveno and David Schwan might better use their time programming around the insufficiencies of their particular BASICs than cry out against an unjust God or a crooked salesman who hooked them on the particular software they have.

"The enclosed subroutine (see Program 1) for ARCSIN contains only five statements. With a few more statements, it can be expanded to give ARCCOS and ARCTAN. If your BASIC doesn't include SIN and COS maybe you need a new assembler. Good luck!" Note that Mr. Ayers' program uses an iterative technique that will be of some interest to those in our readership more mathematically inclined.

In many BASICs, arc tan is already available. Arc cos and arc sin can then be calculated directly as suggested by Gary Marcos, 746 Adams, Albany CA 94706.

"In the November Forum, I read a letter lamenting that there is no arc cos or arc sin function on most BASIC interpreters. This letter was from David Schwan.

"These functions are very valuable for many purposes. One such purpose is for the computation of the distance between two points on a sphere. Recently, I was faced with the problem of converting a FORTRAN IV program, which used an arc cos function, to BASIC. Fortunately, most BASIC interpreters do have an arc tan function to define arc cos and arc sin. I'm dealing with radians not degrees (per se), but the conversion's easy (Example 1)."

```
1000 Y=.2345
1001 REM   SUBROUTINE FOR   X=ARCSIN Y
1002 X=Y*1.5708
1004 D=Y-SIN(X)
1006 IF ABS(D)<.000001   THEN 1010
1007 X=X+D/COS(X)
1009 GOTO 1004
1010 PRINT X
1011 PRINT SIN(X)
1012 END
```

*Program 1.*

```
10 FOR I=100 TO 999
20 H=INT(I/100)
30 T=INT((I-100*H)/10)
40 U=I-100*H-10*T
50 IF H*H*H+T*T*T+U*U*U<>I THEN 70
60 PRINT I
70 NEXT I
80 END
```

*Program 2.*

```
100 S=TI
110 FOR A=1 TO 9
120 FOR B=0 TO 9
130 FOR C=0 TO 9
140 IF A*A*A+B*B*B+C*C*C=100*A+10*B+C THEN PRINT 100*A+10*B+C
150 NEXT C
160 NEXT B
170 NEXT A
180 PRINT (TI-S)/60
190 END
    RUN
    153
    370
    371
    407
    26.47.......
```

*Program 3.*

Let $\theta$ be the angle to be computed (radians).
Let A be the value given.
Then to compute: $\theta = \text{arc cos (A)}$.
If A = 0 then $\theta = 1.570796327$

If $A > 0$ then $\theta = \text{arc tan} \left(\frac{\sqrt{1-A^2}}{A}\right)$

If $A < 0$ then $\theta = \pi - \text{arc tan} \left(\frac{\sqrt{1-A^2}}{A}\right)$

For arcsin: $\theta = \text{arc sin (A)}$
If A = 1 then $\theta = 1.570796327$
If $A < 0$ then $\theta = -\text{arc tan} \frac{A}{\sqrt{1-A^2}}$
If $A > 0$ then $\theta = \text{arc tan} \frac{A}{\sqrt{1-A^2}}$

*Example 1.*

We wish also to acknowledge two letters containing essentially the same information as Gary's. These were from Jon Kapecki, 100 Avondale Pk., Rochester NY 14620, and Phillip O. Martel, 100 Plastics Ave., Rm 2279, Pittsfield MA 01201.

## Programming Problem Solutions

The response to our December programming problem was overwhelming. From the comments of those participating, it is clear that we are providing many readers with the challenge they need to dig into and learn something about BASIC. We, too, have learned a few new wrinkles while examining the many programs received. We regret that we cannot publish every one since space does not permit. A few selected entries will help illustrate the various methods employed to obtain the solution. For those who may not be familiar with the December problem, it was stated in this way: "Write a program to find all three-digit numbers for which the sum of the cube of the digits is equal to the number."

The solutions received fell basically into one of three main types, which we will denote as methods A, B and C. A brief description of each follows.

*Method A.* All numbers between 100 and 999 are mathematically disassembled into their component digits, then tested for the condition stated.

*Method B.* A three-nested loop is used to test the digits; then, if the condition is met, the three-digit number is assembled from current loop values.

*Method C.* Usually a variation of B in which certain values are precalculated and stored to increase execution speed.

An example of each method will help illustrate. Method A is shown in Program 2. Line 10 sets up a FOR-NEXT loop to establish all trial numbers. Lines 20-40 disassemble the number I into its H (hundreds digit), T (tens digit) and U (units digit). Line 50 tests the conditions of the problem and, if not true, skips the PRINT I statement. Note in this line that

```
10 DEFINT A-Z
20 FOR I = 0 TO 9:   CU(I) = I  3: NEXT
30 FOR I = 1 TO 9
40 FOR J = 0 TO 9
50 FOR K = 0 TO 9
60 IF 100*I+10*J+K = CU(I)+CU(J)+CU(K) THEN PRINT I;J;K
70 NEXT K
80 NEXT J
90 NEXT I
The solution set is  1  5  3
                     3  7  0
                     3  7  1
                     4  0  7
```

*Program 4.*

```
110 DEFINT A-Z
120 FOR I = 0 TO 9: CU(I) = I  3: NEXT
130 FOR I = 1 TO 9
140 FOR J = 0 TO 9
150 N = 100*I+10*J: S = CU(I)+CU(J)
160 IF N < S THEN 220
170 IF N > 2*INT(N/2) THEN 210
180 FOR K = 0 TO 9
190 IF N + K = S + CU(K) THEN PRINT I;J;K
200 NEXT K
210 NEXT J
220 NEXT I
```

*Program 5.*

multiplication was used to cube the digits because exponentiation using the ↑ would possibly have introduced round-off error. Method A is the most obvious approach, but not necessarily the best.

Method B was used by Terrell D. Abendroth, 3249 D. Street, Fort Sheriden IL 60037. He writes: "This program was run on a Commodore PET 2001 having 8K BASIC. This 6502-based system has a real-time clock (TI$ gives hours-minutes-seconds; TI gives elapsed time in "jiffies"—1/60 second), so I made the program time itself (steps 100 and 180). Because exponentiation uses logarithms, a small rounding error sometimes occurs. Normally, this would be of little consequence, but it does affect logic decisions about equality. For that reason step 140 uses successive multiplication instead of exponents . . . ran in 26.5 seconds.

"Your column is an excellent means of learning a wide variety of problem-solving approaches. The series of problems you are presenting is a great incentive to get actively involved in efficient program writing."

Line 140 of Program 3 assembles and tests the digits generated in the nested FOR-NEXT loop. Method B seemed a little faster than method A, but it was difficult to be sure because so many different programs and machines were in use.

Of those who used method C, we picked a letter from Jack Thompson, Information Processing Systems, Memphis TN 38122. "Here are a couple of solutions to the problem presented in the December Forum. The first, Program 4, uses the brute-force method. After finding the cube of the digits 0 to 9 and assigning each of these to elements of a vector CU(0) to CU(9), it simply checks all 3-digit numbers against the sum of the cube of their digits and prints those that are equal.

"In the second, Program 5, a couple of additional tests are included to eliminate testing of some of the numbers. In line 160, a test is made to see if the sum of the cube of the first two digits is already greater than the number to be tested. If so, then there is no need to add the third digit to the number. For example, if the first digit, I, is 1, and the second digit, J, is 6, then the test checks to see if 160 is less than $I^3 + J^3$ (=217). If the test is true, then we may eliminate all further 3-digit numbers beginning with 1 because any further increase in J will increase $I^3 + J^3$ faster than it will increase 100I + 10J.

"The test in line 170 uses the following reasoning: The cube of an odd number is odd; the cube of an even number is even. Suppose we are testing an odd 3-digit number and the sum of the cube of the first two digits of this number is odd. Then, adding the cube of the third (odd) digit to this sum will produce an even result which, of course, could not equal the odd 3-digit number we are testing. Thus, if we are testing an odd 3-digit number, then the sum of the cube of the first two digits cannot be odd.

"Now suppose, instead, that we are testing an even 3-digit number and the sum of the cube of the first two digits is odd. Adding the cube of the third (even) digit to this sum would produce an odd result that could not possibly equal the even 3-digit number we are testing. Thus, if we are testing an even 3-digit number, the sum of the cube of the first two digits cannot be odd. Between the two cases, we may eliminate all numbers where the sum of the cube of their first two digits is odd.

"Out of curiosity, I ran this last program on the Xerox Sigma 9 at Memphis State University using Xerox Extended BASIC in a run time of *0.51 seconds!* I hope you continue these little programs in the future. They can be quite fun."

As in a previous Forum we have resorted to a table to summarize the many solutions received. Included with each entry are these data items: name and address of programmer, computer used, method and run time. The entries are given for convenience in alphabetical order. While we call this BASIC

*Table 1.*

| Name | System | Method | Time |
|---|---|---|---|
| Darel D. Eschbach<br>Arizona State University<br>Tempe AZ 85281 | PDP-11/70 | B | 6 s |
| Jack R. Frank<br>638 W. Addison #24<br>Chicago IL 60613 | TRS-80<br>IBM 370/158<br>(FORTRAN) | A<br>A | ? s<br>.21 s |
| Jim Gammell<br>425 So. Oly #13<br>Kennewick WA 99336 | PT SOL 20<br>5K BASIC | B<br>C | 38 s<br>18.4 s |
| Clive Grant<br>Compumatrix Inc.<br>Airport Rd.<br>Laconia NH 03246 | Honeywell 635 | B | .475 s |
| Rodney V. Hamilton<br>29 North Alder Dr.<br>Orlando FL 32807 | SWTP<br>8K BASIC | C | 48 s |
| John E. Hartford<br>50 Maple Sq.<br>Franklin NH 03235 | TRS-80 | B | 38 s |
| Joe Holliday<br>Box 1<br>Luverne AL 36049 | HP 2000 | A | 13 s |

| | | | |
|---|---|---|---|
| David Husnian<br>1731 NW 29<br>Oklahoma City OK 73106 | DEC BASIC PLUS | B | 2.8 s |
| Thomas E. Hutchinson<br>35 Warrender Ave. Apt. 208<br>Islington, Ontario M9B 5Z5<br>Canada | TI-58 | A | 1½ hrs |
| F. Robert Jacobs<br>3013 Trentwood Rd.<br>Columbus OH | SWTP 6800<br>8K BASIC | C | 51 s |
| Mark R. Kato<br>1114 W. 123rd St.<br>Los Angeles CA 90044 | WANG PCS | A | 163 s |
| J. E. Kircher<br>2301 Palmyra Rd.<br>Hannibal MO 63401 | Digital Group<br>Z-80<br>P.T. 5K BASIC | A<br>C | 63 s<br>14 s |
| Michael C. Koss<br>1534 NW 31st St.<br>Oklahoma City OK 73118 | Apple-II | A<br>B<br>C | 17 s<br>13 s<br>8 s |

Forum, you will note that we included programs run in other languages and on hand calculators.

Although generalizations are difficult to make and sometimes hazardous, there is one in programming that is widely accepted. Stated simply it is, "Run time varies inversely with memory space used by the program." In other words, methods that speed execution generally use more memory. In many cases, the programmer merely exchanges slower mathematical calculations for faster data manipulation in memory. So long as memory is not at a premium, the speed advantage should be taken.

The December problem seems to support this idea. Methods A and B use the least memory, depending as they do on brute-force calculation. As expected, they give the slowest execution speeds. Method C, on the other hand, uses more memory in the form of array storage, thus avoiding much repetitive calculation. The result—better execution times. Of course, a terribly inefficient algorithm using huge blocks of memory could be made that would be as slow as *next* Christmas! Perhaps that's why our English teacher used to admonish us that "a generalization is not worth a damn!"

The past few programming problems have emphasized calculation. We thought for a change we would submit a data-manipulation puzzle to readers of the Forum. This program has a way of being deceptively simple to beginners—so watch out!

### Casting Out Duplicates

Write a BASIC program (1) that will accept any list of integer numbers of three digits or less, then (2) print the entire list as entered, then (3) reprint all elements of the list that appear only once (see below).

List: 12, 36, 4, -8, 12, 4
Print: 36, -8

Use the following list below as a test for your program: 6, -10, 15, 7, 7, 7, 6, -8, 7, 2, 150, -6, 13, 12, 12, 5, -5, 19, 18, 19, 18, 19, 105, 4 21, 31, 5.

Try to make your solution program as memory efficient as possible. Assume that the list will contain fewer than 100 items.

Send your solution and any comments to The BASIC Forum, PO Box 7082, Tyler TX 75711. (Note this address. Please do *not* send BASIC Forum-related material to Peterborough. Thank you.)

*John and Dick include in this month's Forum a list that contains the names and addresses of some of those who submitted results obtained from running the December problem. The list includes the type of system used, the method and the run time. These ranged from an IBM 370/158 with a run time of .21 seconds to a TI 58 programmable calculator with a run time of 1½ hours (see Table 1).—John.*

# BOOKS

*An Introduction to Microcomputers, Vol II*
**(June 1977 Revision)**
**Osborne, Jacobson, Kane**
**Osborne and Associates, Inc.**
**Berkeley CA**
**1176 pages, $15**

How do you review a book like this? You could go on for pages about the history of its first edition; the way its success shook up the book publishers ("What? 30,000 copies sold in the first five months? There must be a huge market out there!"); the way it signaled a mad rush to bring out new products (can't you just imagine a group of bleary-eyed toy designers leafing through their well-worn copies of Osborne's first book, trying to figure out which chip to order?); the way it sold to a much larger audience than the author anticipated (it is a standard feature of hobbyists' libraries, used for college courses, skimmed by managers, as well as being indispensable to design engineers).

Or you could discuss whether his offhand comments (more specifically, the benchmark program Osborne used in his first edition—as well as here) have had any influence on the design of more recent chips . . . or you could trace the revisions, expansions, revisions, deletions, revisions, etc., that Osborne and his crew have undertaken to produce this hefty 1176-page volume from two chapters totaling 151 pages in the first edition. But instead of all that . . . let me try to describe this volume as it stands now, without mention of its past history.

Although a wide range of people will find this book interesting and fun to read or skim, it is really aimed at a very specific group—people who are in the process of choosing which microprocessor to use in a specific application.

If I were in that situation, I would want the chance to sit down with an expert to chat about what's available, compare alternatives, suggest relevant criteria for selecting one chip over another and so on. In addition, I'd want to have spec sheets from each chip, including descriptions of the instruction sets. This is exactly what this book provides—just about everything you'd need to know, except prices.

Specifically, this latest revision covers the four-bit single-chip TMS 1000 series of microcomputers by Texas Instruments, the Fairchild F8, National Semiconductor's SC/MP, the 8080A, Intel's 8085, the Zilog Z-80, the 6800, MOS Tech's 6500, the Signetics 2650, the COSMAC, the IM6100, the SMS300 microcontroller, the Pace, General Instruments' CP1600, TI's 9900, two different single-chip micro Novas, plus shorter sections on three different lines of bit-slice products and an overview of the Hewlett-Packard MC2 microprocessor.

Appropriately, the 8080A chapter is the longest and, where reasonable, other products are compared to the 8080 and 8085. In most cases, the description of the microprocessor is followed by descriptions of relevant support chips. While some of the included material is taken directly from manufacturer spec sheets, the authors attempt to describe each chip in a uniform language and notation so the reader doesn't get lost in conflicting terminology.

There is something distinctive about the writing style that I can't quite put my finger on. It's straightforward, not "shooting from the hip," and very decisive—not exactly humorless as much as sincere. It's as if Osborne himself, half computer expert, half private eye, is sitting on a stool across from you, smoke swirling in the bright light. He speaks in short, sharply pointed sentences. He doesn't want you to go astray.

He is supremely sure of his motives, ethics and methods, even though the world is a sticky place. ". . . instruction sets are very subjective; right and wrong, good and bad are not easily defined." When he has the facts to back him up, he pulls no punches. This book delivers.

**Rich Didday**
**Santa Cruz CA**

---

*Programming Proverbs* **and**
*Programming Proverbs for FORTRAN Programmers*
**Henry F. Ledgard**
**Hayden Book Company, Inc.**
**Rochelle Park NJ**
**1975, $6.95**

---

Except for the program examples, these two books are almost identical, word for word, so there is no need to buy both. The shared content, however, is so useful that I recommend getting one of them. The programs in the first book are written mostly in PL/1 and ALGOL 60 with a smattering of BASIC, while the latter book gives most of its examples in FORTRAN. Although I generally program in FORTRAN on big machines, I found the first book more interesting because of its variety. Knowledge of the language used in the examples was helpful, since I was in the midst of examining a lot of languages and was able to follow the examples.

The highlights of the books are the 25 proverbs that form chapter two and the emphasis on top-down programming throughout the books. They detail an extremely common-sense and logical technique for doing any kind of programming. Some of the suggestions may initially offend some programmers who pride themselves on being able to write instant code or compact programs into a few lines. Based on my experiences before and after reading the books, programs are a lot easier to develop and, especially, to come back to if the techniques are followed.

Top-down programming is essentially the process of defining the problem several times, each time in more and more detail. Each definition serves as a guide to find the next solution. Furthermore, the process emphasizes constructing a series of modules, which I have often found useful in other programs.

Example proverbs include: (#2) *think first, program later . . .* "Examine the problem carefully. Consider alternative approaches . . . Give yourself time to polish the algorithm." (#12) *use intermediate variables properly.* The first example illustrates how a lack of intermediate variables can bury the outline of the program. The second example displays the outline more clearly.

```
RESULT = ALOG(SQRT(EL-2.0*FULL(R-Y)))+4.0*FULL(Y-R)
```

*Example 1. Lack of intermediate variables.*

```
WEIGHT=ALOG(SQRT(EL-2.0*FULL(R-Y)))
     SIZE=4.0*FULL(Y-R)   COST=WEIGHT+SIZE
```

*Example 2.*

The books are written in a style that is fun to read. For those who feel terribly bound, proverbs #24 (consider another language) and #25 (don't be afraid to start over) can be quite relaxing. The balance of the book includes some thoughts about specific programming problems and expansion on details of several of the proverb topics, including mnemonic names, prettyprinting and recursion.

I recommend these books, which are available in many computer stores and some libraries, as well as from the publisher. The FORTRAN book, with its bright pink cover, particularly stands out on the store shelf. It is worth getting beyond the cover.

**Mike Firth**
**Dallas TX**

---

*Stimulating Simulations*
**C. William Engel (author-pub.)**
**Tampa FL**

---

*Stimulating Simulations* is a collection of ten programs written for the computer buff who has just progressed beyond the simple number-guessing games and is ready for a little imagination. At first glance the book appears to comprise superfluous information, but I am sure that the novice programmer will appreciate the explanations and flowcharts. Each simulation contains a scenario, sample run and flowchart, followed by a description of the variables, then the listing.

The simply written lists make modifications to other systems easy. Many lines contain only one statement, and are numbered in multiples of ten. All REMark statements have a units digit of five, and the rules are written in the third person for placement in a subroutine. Suggestions for program modifications are given to spur the reader's creativity. In some cases, formats for playing boards, charts and graphs are supplied.

Some of the simpler simulations are Monster Chase and Art Auction, in which the skill of the operator is tested mildly (trying to elude the monster's clutches for ten moves can sometimes be difficult!). Gone Fishing, Space Flight and Forest Fire are rather routine, but offer languid entertainment to someone trying to outwit the computer. The most complex and interesting is Diamond Thief, where you, as detective, try to determine which of five suspects is the culprit. Your task is complicated by suspects having a five-percent chance of error and a like chance of forgetfulness. The whole run can be much more fun than the old board games.

In general, Dr. Engel's simulations show reasonable imagination without the complex routines commonly found in programs of this type. *Stimulating Simulations* should be useful to the beginner because it gives detailed instructions and does not require extremely advanced BASIC commands. Once you're into them, however, the ten routines go fast, and you will soon be looking for a more advanced edition.

**Robert Soltysik**
**Plano TX**

---

*IC Timer Cookbook*
**Walter G. Jung**
**Howard W. Sams & Co.**
**Indianapolis IN**
**287 pages, $9.95**

---

I have noticed in several places statements that suggest the IC timer is as important and useful as the op amp. Here is a book that proves this by providing numerous circuits, and also puts a lot of information on the 555 and its relatives in one place. Although I have seen a lot of different applications for the IC timer, there were numerous ideas presented in this book that I had not yet come across. I believe that this is because many of these circuits have come from professional magazines such as *Electronics.*

If you have seen Walter Jung's other book, *The IC Op Amp Cookbook*, you will find the layout of this book familiar. The book leads off with a description of the basic RC timer, around which all IC timers revolve. Now the reader is ready to discover the workings of specific IC timers, including the 555, 556, 322, 3905, 2240, 2250 and 8260. I am sure that everyone is familiar with the 555 and 556 but, you might ask, what are these other ones? They are precision and programmable timers. (This is not the place to get technical, so either get someone to write about these for *Kilobaud* or get this book if you want to know more.)

The book's second chapter includes block diagrams, internal schematics and pin-by-pin descriptions of the devices. I have noticed that people miss a lot because they are not properly acquainted with the full capabilities of some ICs; so this information is very helpful.

The third chapter is devoted to general information about IC timers. Included here are pin connections, design precautions and some thoughts about components to be used in conjunction with the timers.

With the basics behind him, the reader of this book is now ready to enter the realm of actual applications. The applications section is broken down into three chapters: "Monostable Timer Circuits," "Astable Timer Circuits" and "IC Timer Systems Applications." Circuits here range from an astable that uses only one resistor and one capacitor, to a "Wide Range Pulse Generator." Full information is provided along with the

*(continued on page 21)*

# NEW PRODUCTS

### ANSI Standard FORTRAN IV

Technical Design Labs announces the first complete ANSI Standard FORTRAN IV for a microcomputer, written for Technical Design Labs and the Z-80 by Small Systems Services, Inc.

Operationally, this FORTRAN is a disk-oriented system. It runs in less than 24K with DOS, and both FDOS IV and CP/M versions are available.

This FORTRAN IV package includes both the floppy diskette with object code and a user's manual. Additional documentation and support packages are available. It is priced at $349.

Technical Design Labs, Inc., Research Park, Building H, 1101 State Road, Princeton NJ 08540.

### New Drop in Memory Prices

The new refresh design, SynchroFresh, is simpler than previous approaches. SynchroFresh-equipped 8K memories have been announced as low as $149. Using SynchroFresh, the new 8K memories use half the power of static boards, and can undersell both static and older design dynamic memories.

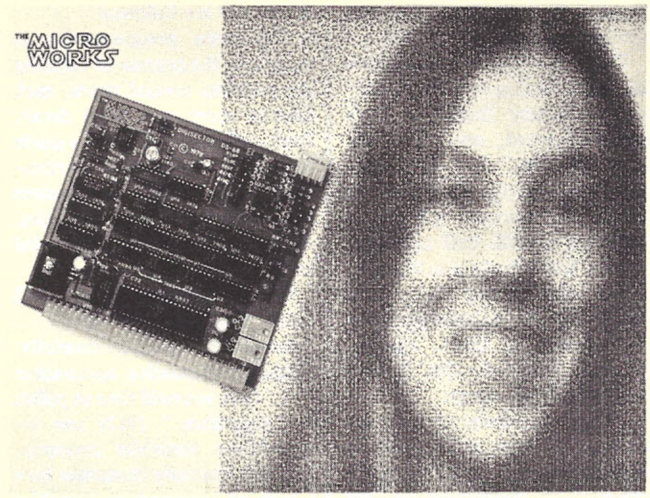
The SynchroFresh system eliminates reliability problems because it does not interrupt normal CPU operations or timing in order to perform memory refresh. Instead, inventor/designer George Morrow planned SynchroFresh to utilize the natural timing of the S-100 bus. SynchroFresh circuitry monitors the microprocessor's machine states, utilizing the $T_4$ states for refresh. $T_4$ always occurs during instruction fetches, leaving memory available for refresh.

The Thinker Toy Econoram III 8K with SynchroFresh is being supplied as part of The Equinox personal computer system by Godbout Electronics, and is available by direct mail from Thinker Toys.

Thinker Toys, 1201 10th St., Berkeley CA 94710.

### The Micro Works Digital Video System

The Micro Works Digisector (DS-68) allows a 6800 computer system to see! The Digisector functions with an inexpensive television camera to present the computer with a high-resolution digitized picture. The DS-68 requires one I/O slot in the SWTP 6800 computer (or equivalent) and accepts either interlaced (NTSC) or non-interlaced (Industrial) sync pulses from the video source. It features 256 by 256 picture element resolution, with up to 64 levels of grey scale. Data conversion times can be as low as three microseconds per picture element. (The computer portrait shown in the picture was taken by a DS-68 and printed on the Malibu Design Group's Model 160 printer.)



*The Digisector meets the Malibu Design Group's Printer.*

Operation is simple. The computer sends the DS-68 two 8-bit addresses (X and Y coordinates), and it returns the digitized brightness of the image at the specified location. Applications include precision security systems, moving target indicators, computer portraiture and more. With cleverly written software, the DS-68 can read paper tape, punched cards, strip charts, bar codes, musical scores and *Kilobaud*.

Like all Micro Works products, the Digisector comes fully assembled, tested and burned in. The price is $169.95; software for computer portraiture and slow-scan television is included.

The Micro Works, PO Box 1110, Del Mar CA 92014.

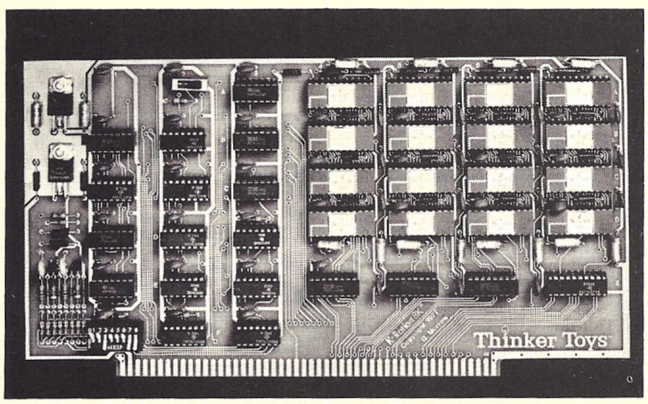
### CRT by North Star

North Star Computers, Inc., manufacturer of the HORIZON computer, now offers a 24 line by 80 character CRT display terminal for use with the HORIZON computer. The CRT terminal, manufactured under agreement with SOROC Technology, can be connected to the HORIZON with I/O port at baud rates up to 9600 baud. A 90-day limited warranty is honored by SOROC.

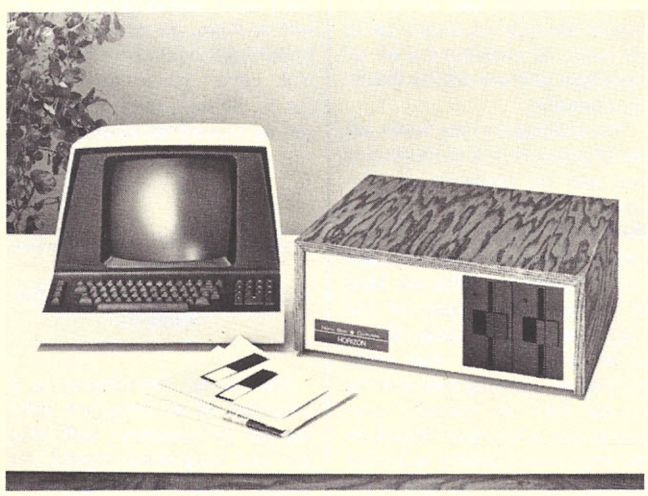
The HORIZON is a complete, disk-oriented computer with a 4 MHz Z-80A processor, 12-slot S-100 motherboard, 16K byte RAM, one or two Shugart minifloppy disk drives and a standard serial I/O interface. Expansion to three drives and more than 64K RAM is possible. A version of North Star's extended disk BASIC is included with each HORIZON.

Prices: SOROC IQ 120 Terminal (assembled only) $995; HORIZON-1 (single disk drive) computer: kit $1599; assembled $1995. HORIZON-2 (dual disk drive) computer: kit $1999; assembled $2349.

North Star Computers, Inc., 2547 9th Street, Berkeley CA 94710.



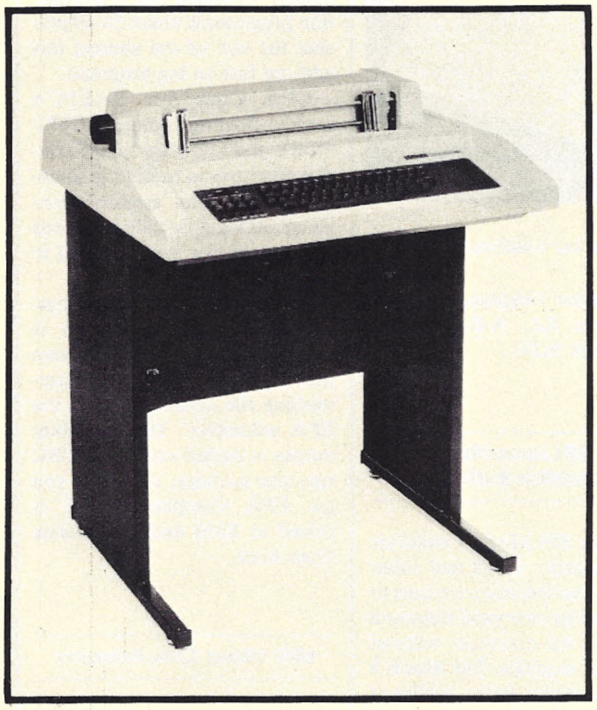

*Thinker Toys Econoram III.*



*HORIZON and CRT.*

# Introducing Intertec's SuperDEC™

**BEFORE**



**The LA-36/DECwriter II**

A 300 Baud teleprinter with *no* features.

**AFTER**

**Intertec's SuperDEC™**

A 1200 Baud teleprinter with *many* features.

# Can you see the $395 difference?

While we'll admit the difference in appearance between the DECwriter II and our new SuperDEC is difficult at best to see, the difference in performance is astounding! The SuperDEC is our new Throughput Optimizer designed to be easily installed in your existing DECwriter II teleprinter. Not only can our SuperDEC Optimizer increase the print speed of your DECwriter II by as much as six times its original speed, it also gives you the features offered only by our famous SuperTerm teleprinter. Features you couldn't get on your DECwriter until now.

You've undoubtedly already heard of our SuperTerm. It's the 1200 baud teleprinter that has been replacing DECwriters by the thousands. And while you may have purchased your DECwriter prior to the introduction of our state-of-the-art SuperTerm, you can now have all of the SuperTerm's incredible features without having to throw out your DECwriter.

For just $395 you can throw out the guts of your DECwriter and install the brains of our SuperDEC Throughput Optimizer. The SuperDEC Optimizer is designed to replace the digital electronics in your existing DECwriter II. In less than five minutes, your DECwriter can be transformed into a SuperDEC. The SuperDEC Optimizer is completely "plug-compatible" with the cables in your DECwriter. The only installation tool required is one that we give you—a screwdriver. Just pull out the guts and screw in the brains. No special technical skills are required. And if you get bored watching your DECwriter print faster than you can read, the old digital electronics may be reinstalled in a matter of minutes. It's really just that simple.

While speed will be the most obvious personality change in your DECwriter when equipped with our SuperDEC Optimizer, there are many more subtle changes you will begin to notice.

With the SuperDEC Optimizer installed, you will have such nifty features as bi-directional printing, manual and automatic top of form, full horizontal and vertical tabs (addressable and absolute), adjustable right and left margins, an RS-232C interface, a double wide character set and up to 32 user programmable characters. You can also add an APL character set, selective addressing and an answer back feature at nominal cost.

Every SuperDEC Throughput Optimizer carries a full one year warranty on all parts and workmanship. But our commitment to excellence in service goes beyond the warranty. Intertec can also offer on-site service contracts for all of your upgraded SuperDEC equipment.

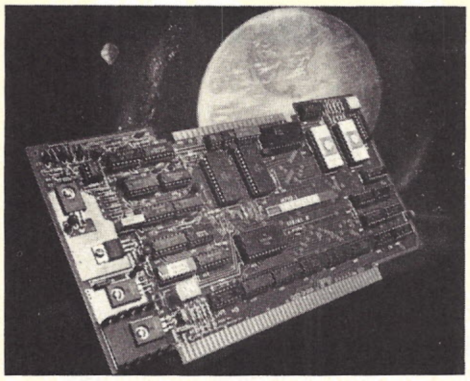
So, when you're ready to "pull out the guts and screw in the brains", contact us at one of the numbers below and we'll give you the name of your nearest SuperDEC dealer. He'll show you what a difference $395 can make.


**INTERTEC DATA SYSTEMS** 121

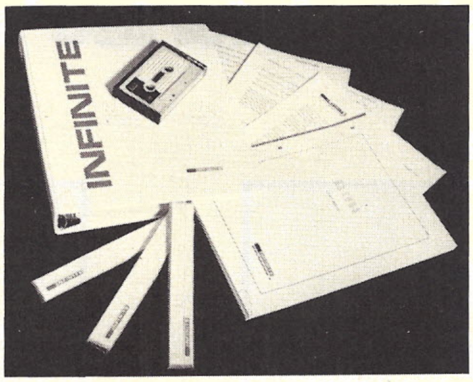
**Corporate Headquarters**
1851 Interstate 85 South
Charlotte, North Carolina 28208
704/377-0300

**Eastern Regional Marketing**
19530 Club House Road
Gaithersburg, Maryland 20760
301/948-2400

**Western Regional Marketing**
17952 Sky Park Blvd.
Irvine, California 92714
714/957-0300



*Infinite's MFIO-1.*




*Some of Infinite Inc's software.*

### Infinite Software and New I/O Board

Infinite Incorporated, 1924 Waverly Pl., Melbourne FL 32901, is making available a wide assortment of low-cost software for the COSMAC 1802 microprocessor. This assortment includes a range of levels from machine language to BASIC, a variety of applications from mathematics to music and a selection of media from listing to cassette cartridges. All software packages will include comprehensive user instructions.

Infinite publishes a software library list that is updated monthly and contains a description of all packages released to date. The company also designs and markets the UC1800 series of 1802-based microcomputers.

* * *

Infinite also announces the first in a series of 8080-oriented products, the MFIO-1, an S-100-compatible general-purpose I/O board containing a major portion of all circuitry required for a complete microcomputer.

The product comes in three versions—assembled and tested ($282), complete kit ($234) and bare boards ($49). Set of 2 ROMs, $65.95.

### Vector Graphic Introduces Bit Streamer I/O Board

Vector Graphic's Bit Streamer design concept combines two parallel input and output ports, and a serial I/O port using an 8251 programmable USART. Communications with board circuitry is accomplished by the CPU. One parallel port also can be used as a keyboard input port. The USART is designed to interface easily to an S-100 bus structure and is capable of being configured for a wide variety of communication formats.

The Bit Streamer, priced at $155 kit, $195 assembled, has been designed for ease of construction. Without changes to the pre-jumpered options, the board can be installed in a computer and will operate as an RS-232 serial port using the initialization and I/O software on the Vector Graphic option C PROM.

Technical data covering the "Bit Streamer" I/O board and other products may be obtained from Vector Graphic, Inc., 790 Hampshire Rd., A-B, Westlake Village CA 91361.

### BPI Intensifier Multiline Buffers

A single BPI Model 8 multiline buffer permits CRTs and other RS-232C compatible terminals to be located up to several thousand feet from the computer without the use of modems. The Model 8 includes eight fully buffered lines; the Model 18 includes 18 fully buffered lines.

Single-quantity price for the Model 8 is $149—$46 more for the Model 18. All units carry a full-year warranty.

BPI Electronics, Inc., 4470 S.W. 74 Ave., Miami FL 33155.

### EPA Compiler BASIC

Electronic Product Associates, Inc., 1157 Vega St., San Diego CA 92110, announces the new EPA Compiler BASIC. You can use it to build business applications, with decimal arithmetic for penny amounts up to $99,999,999.99 formatted output, strings and multiple disk file I/O. Long variable names aid program maintenance. Packaged applications can't be stolen because you don't need to sell the source. Compiled size of application programs is about 50-60 percent the size of the source; this adds up fast on big programs.

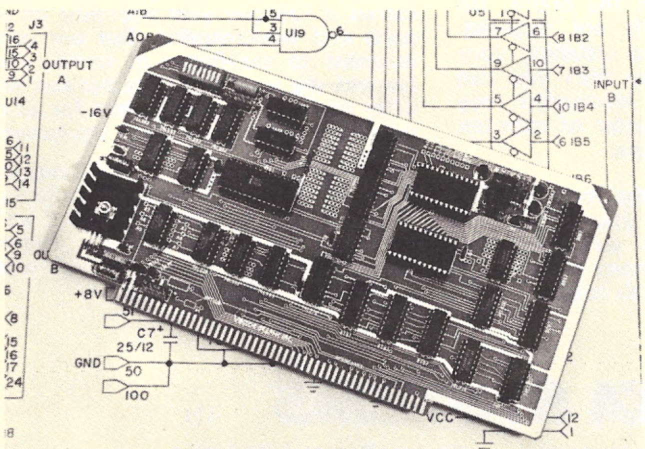
EPA Compiler BASIC's speed, floating point, PEEK/POKE and I/O allow many control programs to be built in something other than assembly language. ROM-able code generated by the compiler can be placed in your micro and forgotten.

Program generation uses whatever text-processing system is available. The compiler processes this text and produces an intermediate file assembled using the EPA assembler. The assembler output is loaded with the BASIC run-time package, and away you go. EPA Compiler BASIC is priced at $250 and is available from stock.

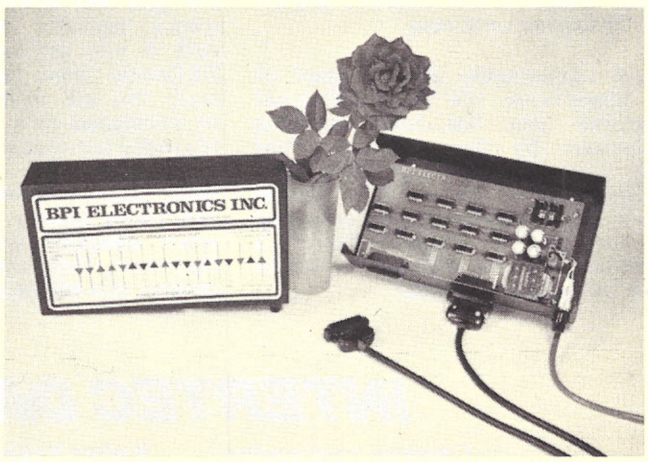
### 6800 Object Code Relocator

Technical Systems Consultants, Inc., PO Box 2574, W. Lafayette IN 47906, now has a machine-code relocator for the 6800 microprocessor. This program gives you the capability of moving assembly-language programs from one area in memory to another. A special feature is included that allows loading a Motorola MIKBUG format tape directly into any part of RAM. This means programs located on tape where no RAM is available may still be loaded.

Use of the relocator requires a knowledge of where the program to be moved starts and ends and all places in the program that contain data as opposed to ex-



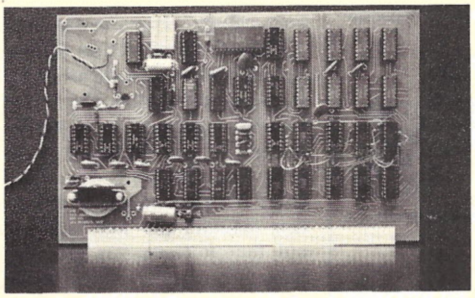

*The Vector Graphic Bit Streamer.*



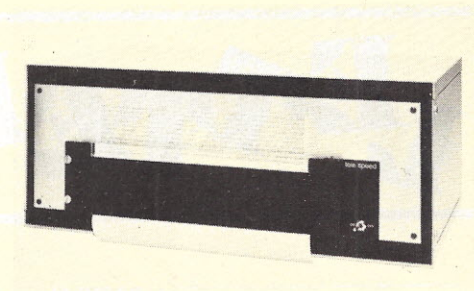

*BPI Intensifier.*

ecutable code. All references to locations outside a range specified by the user will be left unchanged so that calls to monitor routines or other external routines will be properly relocated.

The price of $8 includes a commented source listing, object code listing and a comprehensive user's manual giving several samples of use of the package.



*VDB-1 Video Display Board.*



*Tele Speed Model 81 Printer.*

### Compact New Power Supply

Forethought Products, maker of the KIM to S-100 interface/motherboard "Kimsi," has announced the new Kimsi-Plus Power Supply, housed in a single high-quality unit. Designed specifically to power a full Kimsi system (including KIM, Kimsi, and eight S-100 boards), it could also power any S-100 system with 8 to 10 motherboard slots.

Measuring 8½ x 4½ x 5½ inches, the supply is mounted on a heavy-gauge open-frame type chassis, which allows either built-in or stand-alone operation. Its 16 Amp transformer and 30 Amp rectifier allow the unit to deliver full output without forced air cooling, which other high-output supplies may require.

The Kimsi-Plus Power Supply is available for $69.50 kit or $89 assembled.

Forethought Products, PO Box 8066, Coburg OR 97401.

### ACI-33 Cassette Interface

The ACI-33 is a simplified audio cassette interface designed primarily for the SWTP 6800, the control interface and a terminal. The unit will also operate with any RS-232 terminal and computer serial I/O that can supply +5 V, and ±12 V for the RS-232 interface. When used with the SWTP, the ACI-33 supports all functions of the control interface, including loop-current teleprinter applications.

The ACI-33 uses the self-clocking redundant Manchester scheme of encoding, sometimes called Kansas City Standard. The two logic states are represented by a specified number of cycles of 1200 Hz and 2400 Hz, which are precisely written and read from the tape.

To use the interface, it is only necessary to plug it into an unused I/O slot on the motherboard (for power), plug the terminal that was connected to the control interface into the connector provided on the ACI-33, and the connector from the ACI-33 cable into the control interface connector. The audio cassette recorder, Auto-Manual switch and data indicator are connected to another connector provided on the top edge of the printed circuit card.

The LED Data indicator shows the presence of carrier and data. The switch is used to provide the signal to the data path control circuit to choose either data from the terminal or data and clock from the tape or auto computer control. These controls can be mounted remotely at the terminal or recorder for convenience. Price, $59.95.

Personal Computing Company, 3321 Towerwood Drive Suite 101, Dallas TX 75234.

### Circuit Board for VDB-1

F&D Associates have arranged with Alfred Anderson to supply a printed circuit board for his VDB-1 Video Display Board. F&D's board is plug-in compatible with the SWTP 6800. It is also compatible with any 6800 or 6502 based uP. Display format is two pages of 16 lines x 32 characters. Software is included for scrolling, screen erase, etc. The board has provisions for Pixieverter or direct video, and on-board regulation. The bare VDB-1 board, software and documentation is $29. Add $2.50 per order S/H. (Documentation only, $5 postpaid; refundable with order.) Ohio residents, add 4 percent tax. F&D Associates, 1270 Todd Rd., New Plymouth OH 45654.

### New Tele Speed Printer

Tele Speed Communications, Inc., PO Box 647, Syosset NY 11791, is offering a new, inexpensive dot-matrix serial-impact printer.

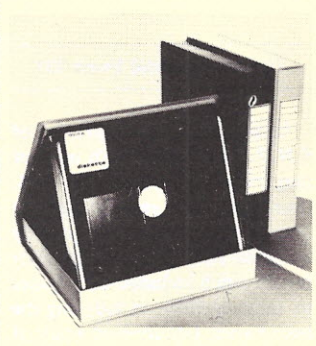
The Model 81 Printer is an 80 cps, 80+ column, bidirectional, asynchronous printer, complete with electronics, power supply and cabinet. The printing medium is friction-fed pressure-sensitive paper. A ribbon mechanism and a tractor mechanism are optional.

The unit's paper advance and carriage are stepper motor driven permitting the unit to be used for graphics or as a plotter under microprocessor control.

The Model 81 Printer with parallel ASCII interface is $615.

### Organized Protection for Diskettes

Alpha Supply Company announces the KAS-ETTE/10 Library Case, which provides an ideal way to handle diskettes while in use, permanently store diskettes or safely ship several diskettes. The case is made of durable molded plastic and looks like a leather-bound book—available in blue or beige.

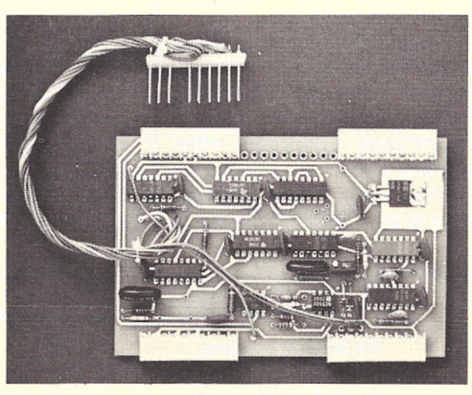


*The KAS-ETTE/10—open and closed.*

When open and in use, a molded plastic insert provides pop-up convenience for locating the desired diskette. Flexible fan tabs hold diskettes securely in an upright position, which assures that diskettes will be protected from warping. When used as permanent storage, the library case protects diskettes against dust and humidity. Color-coded labels applied to the spine of the library case permit users to organize a permanent library.

Alpha Supply Company, 18350 Blackhawk St., Northridge CA 91326.

*Kimsi-Plus Power Supply.*



*ACI-33 Cassette Interface.*

# LETTERS

## MITE Printer Discontinued

"Consider a MITE Printer" by R. W. Burhans (*Kilobaud* No. 11, p. 38) has created quite a furor. As a result of this article, MITE Corp. has been inundated with phone calls from hobbyists all over the eastern seaboard. However, we regretfully inform you that the MITE Printer line has been discontinued and contemplation of reproducing the printers in the future is negative. The residual stock for these printers is currently in the possession of Expandor, Inc., 612 Beaty Road, Monroeville PA 15146, (412) 373-0300.

The MITE Corporation would appreciate your mentioning these facts to your readers.

**Richard A. Ahlers**
**Contract Sales Manager**
**MITE Corporation**
**446 Blake St.**
**New Haven CT 06515**

## Reprint Material From 73?

In my function as Librarian of S.N.P.C.S., I read both of your magazines, *Kilobaud* and *73*, thoroughly and enter articles of interest in our index. We recently accepted your subscription offer for *73* which included back issues to January 1976. In indexing the back issues, I discovered a pair of articles I feel you should consider for publication in *Kilobaud*. I am aware of your policy against publishing the same material in both magazines, but I consider these articles an exception.

I refer to "The Soft Art of Programming," Parts 2 and 3, by Rich Didday in issue 193 and 194 of *73*. I feel they are worthy because of their treatment of external files in BASIC in a microcomputer/audio cassette environment. I am a programmer/analyst with experience on IBM 1401, 360/370, and currently on the General Automation 18/30 minicomputer in Assembler, FORTRAN and RPG-II. My knowledge of disk and tape files is not easily translatable to microcomputer/audio cassette BASIC files, and I am sure there are microcomputer owners with less experience who are in the same boat.

**Cyrus N. Wells, Jr.**
**President**
**Southern Nevada Personal**
**Computing Society**

*We've had a lot of good material in the I/O section of 73 over the last two years, and Rich Didday's series rates as some of the best. It's so good, in fact, that we have* already *reprinted it in* The New Hobby Computers Are Here. *This book is available for $4.95 from* Kilobaud *and contains, in addition to Rich's series, 21 articles on numerous aspects of hobby computing.—John.*

## KIM-1, ACT-1: The Scene

I recently purchased a Micro-Term, Inc., ACT-1 TTY replacement terminal and, after resolving some interfacing problems, I have it running with my KIM-1. Hookup data supplied with the unit is very general and I would like to share my experience with other KIM-1 users.

After making all the external connections and one internal change per the user's manual, I was unable to get the ACT-1 running. I made a few phone calls to Micro-Term, but the results were still negative. The people at Micro-Term, although very cooperative, were unfamiliar with the KIM-1. I finally got up enough courage to experiment. The results that worked are shown in the table.

Part of the confusion comes from the serial output level marking on my board (ACT-1, 4-77, REVD). It is wrong according to Micro-Term. The only other problem was an unsoldered keyswitch. I could not get one character to print. After soldering the connections, everything was fine.

I have the baud rate set at 1200 and have had no problems using the system at this rate. The screen will fill completely in about 20 seconds. I can display a little more than 256 bytes (one KIM page) for each memory dump. This includes the start address and format characters plus the ending line, which uses up some of the space. (My SX70 camera works fine for making a hard copy of the program if I want one.) By setting the interrupt vectors at 17FA-FF to 1C00, I was able to use the ST key to stop the run and examine it at any point. Typing RETURN (after ST) and then Q again when ready started the run at the last address indicated after RETURN was typed. This worked only when the ending address at 17F7-F8 was set at 2000.

Micro-Term has done a good job on the ACT-1, and I recommend this unit to anyone planning to include a serial TVT terminal in his system. I hope that other users derive the same enjoyment from using the ACT-1 that I have.

**Chuck Carpenter**
**Carrollton TX**

| Internal Connection | Connect To |
|---|---|
| Serial Output Level | P |
| Serial Polarity Out(put) | Invert |
| Serial Polarity Input | Unchanged |

## Plea for 6800 Operating System

I first became a reader of your magazine in July 1977, and was so impressed that I simply *had* to order all back issues. There are not too many magazines that I read completely—cover to cover—but yours is one.

That's the good news. Now for the bad! I recently built the Motorola MEK D2 kit and, in the course of familiarizing myself with its operation, became aware of the need for an improved monitor. So . . . I began to read, in depth, all articles dealing with monitor systems in the various magazines in my bookshelf.

The first two issues of *Kilobaud* contain the start of the development of such a monitor (would you believe for the 6800?), which is coming along nicely in issue No. 2. The series, entitled "Practical Microcomputer Programming," is written by John Molnar. At the end of Part 2, he promises that Part 3 will go into his system in some detail—including a listing of his monitor—and there the matter ends. Part 3 merely details comparisons between assembly language, interpreters, compilers, etc.

So here I am, cut off in midstream! Whatever happened to the concluding article? Why did you hold out the promise of such a feast to come, and then, when you had me drooling at the mouth at the thought of all those delicacies, merely serve up hamburger? Without that final article, Part 2 of the series is as nothing . . . like getting absorbed in an exciting mystery novel, only to find that the last 50 pages are missing. You have to get John to write that promised article as soon as possible, before I die of frustration. Here's hoping to see it in print SOON.

**Bob Jones**
**Abbotsford BC**
**Canada**

*We'll try to get our good friend John to put the finishing touches on that project, Bob, but if he can't make it, we've got some similar material in the works that you'll find of interest.—John.*

## A Back Issues Snapper-Upper

I'm writing to express my appreciation of the professional, objective, yet lucid and down-to-earth style of *Kilobaud*. As an interested but bewildered novice, I find most computer magazines abstruse (or obtuse), or philosophically overblown. But *Kilobaud*—ah! I'm snapping up all the back issues I can lay my hands on.

Just finished the November issue, and found out that you had already published a couple of broadsides against problems in the industry that I had been meaning to froth about. I refer, of course, to your articles about salesmanship and advertising by Ken Barbier and Sheila Clarke. Clarke's piece was intelligent, constructive, and precisely to the point. Barbier is far better housebroken than I am.

It is a pity, really, to realize how many little companies are going to go under because of ineffective advertising and lackadaisical sales reps. I think a good many people are in the business because they like computers. It's not enough.

As a consumer, I am eager to buy, but reluctant because I remember the calculator price drops of yesteryear. After seeing the Apple-II, the PET, and the

TRS-80, I wonder what will happen next.

I think a lot of people like me are buying *Kilobaud* and waiting for a big price drop. I think a few words on the subject might have a sizable effect on sales. But, being a novice, I don't really know.

I do know that, like the average guy, I am bored with pictures of little gray boxes and circuit boards, and articles on how to acronym my phase-modulated Macroach to make my BVDs transparent. I am motivated by color and pictures of nice-looking people interacting with computers and enjoying themselves.

Yup, I'm a slob—but not a complete slob (I read *Kilobaud*, don't I?). After I realized that computers were a possible way of expressing human feeling and caring, I realized that I had a solution to a professional problem. So, after getting turned on by the humanistic computer mags, I switched to *Kilobaud* for information and ideas, and to a very different perspective.

I'm a teacher of the deaf. I program computers through defective modems. My debugging and troubleshooting routines would drive a normal programmer crazy. I am good at my work, but I am never going to be good enough. I will snap up anything that makes me more effective; I will spend any amount of my own money—but I will not spend one penny unless I understand a system and know exactly what it can do.

The article about MAXI-Basic (*Kilobaud* No. 10, p. 78) is a case in point. The complaint was valid—it's picayune to complain about a language that suits its function because you're not used to it. All the same, a corporation president has certain responsibilities to his company. He's not supposed to stand up and tell everyone his customers are complaining about his product. He's supposed to jump at the chance to describe his product . . . explain how it is new and powerful and different from anyone else's . . . tell about the wonderful things it can do . . . inform me, impress me, sell me, take my money—and get rich. Just so I get the facts about MAXI-Basic.

With the tremendous information gap between hacker and novice, getting the right slant on editorial and advertising copy is a very tough job. I know your writers grouse about it. Me, I'm no engineer—at least, I haven't noticed any hair growing out of my forehead lately. I'm learning a lot.

You guys are evidently doing something right. I think it's the way you go after the application in nontechnical terms in the first paragraph. Once I know what your doohickey is supposed to do, I get motivated and curious, and I can slog through the heavy stuff. Hook your subject to a human problem right off, and you have me hooked.

**Charlie Heckel**
**Glendale CA**

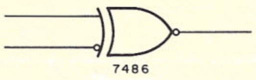
*You mentioned in your opening paragraph that you're a bewildered novice. I've said it before, and I'll say it again . . . to some degree, we're all bewildered novices. That is what* Kilobaud *is all about. (Even if we aren't "bewildered," there are always areas we want to learn about.) You're right about those opening paragraphs. If everyone writing an article would remember how important they are, my job would be a lot easier.—John.*

### Exclusive OR Mismatches?

I am in strong agreement with the ideas expressed by Russell Lauffer in his article on logic diagram conventions in the December 1977 issue of *Kilobaud*. Until about six months ago, I too was naively locked into the practice of drawing gates as they appear in the data books. However, since I started my present job as an engineer for a well-known instrument manufacturer, I quickly learned how much simpler understanding a complex circuit can be using logic *function* drawings.

Russell states that mismatches between inputs and outputs frequently occur when you use flip-flops and XORs. Often, drawing the XOR as shown below will help, which is, of course, logically equivalent to the normal symbol.

**Don Kinzer**
**Portland OR**




### The Systems Selling Game, Revisited

I subscribed to another "experimenter-oriented" computer magazine, and the appearance of *Kilobaud* was like a breath of spring air after a hard, cold winter. Although I've had limited training in FORTRAN and COBOL, my main background is in statistics, accounting and management analysis. I also subscribe to *73 Magazine*, but there seems to be little time to keep up to date on the inner workings of the "black boxes" and circuit boards that make up modern microprocessors. It is from this background that my comments are based.

Two months ago, while in New England, I rented a car and drove to the nearest Computer Retail outlet in order to view first hand some of the various systems. My experiences were not unlike those of Ken Barbier (*Kilobaud* No. 11) —and I too had a substantial sum of money burning a hole in my pocket. It appeared that neighborhood kids playing computer games and previous customers using the salesperson's time to debug programs had priority over new customers. After a two-hour freeway drive, I too was "dying" for a cup of coffee.

Finally, after I persevered (plus was somewhat forceful) for over an hour, a demonstration was arranged. Unfortunately, because the kids still had priority over new customers, the only system "available" for demonstration was the Apple-II. None of the systems that specifically interested me were connected for demonstration—or they were out of stock.

It is worth mentioning that three friends who had no previous exposure to microprocessors accompanied me on this trip. Accordingly, I requested a pep talk and explanation of what microprocessors could do—besides being fancy games machines. It never really materialized. In this situation, a sales pitch and demonstration of applications such as those discussed by Sherman Wantz (*Kilobaud* No. 11) might well have interested my three colleagues. The net result was that I left the store with a brochure on the Apple-II (for further study), and my three companions left wondering who would be crazy enough to spend that kind of money on a fancy games machine.

The above experience served to personally underscore what *Kilobaud* has been expressing for the past few months: Poor salesmanship is losing sales to new potential hobbyists and businesses (my main interest).

In *Kilobaud* No. 13, Wayne Green raised a question about systems for the small business. Having recognized this potential some time ago (given my previously stated background), it was obvious to me that the first requirement for selling a business system is to become a proficient programmer in BASIC, up to and including disk operating systems.

As a starting point, my concept of a small-business system would consist of a video terminal, minimum of 16K RAM, room for possibly another 16K RAM, ability to control two cassettes for external storage/backups, an impact printer, a form of extended BASIC (preferably in ROM) and add-on capability for up to three disk drives. Other desirable features would be ability to accommodate more than one video terminal as an input device (timesharing), and possibly one or more printers (dot matrix acceptable). Finally, 9-digit precision in computations would be desirable if any statistical analysis packages were to be developed for business applications.

I have previously worked in the enhancement and development of large management-information systems. The basic principles that apply—whether selling to large or small business—are succinctly summarized in the opening sentence of Robert Brehm's article in *Kilobaud* No. 13. Ironically, initial sales to businesses are based upon bookkeeping needs—providing timely and accurate financial information—yet, as systems are implemented and accepted by business, the emphasis often shifts to "fringe" benefits such as improving customer service.

For example, with the PAC 1 system described in the above article, it might be desirable to extend the data files to summarize previous patient history, prescriptions, etc., which can be reviewed in the morning for those patients who have appointments that day. This latter comment should not be construed to "pick holes" in a well-written article, but rather to reinforce the depth of analysis and programming needed to sell a good system to a business.

Although some "canned" programs can be mass-produced and sold for small-business applications, it is doubtful that their application will be useful for firms employing more than five people. For larger firms, such as automobile dealers, contractors and retailers, the applications programs will have to be tailored to suit the needs of the particular business. The expertise to accomplish this is unlikely to originate in computer retail outlets as they exist now. More than likely it will originate in persons who have a mixed background in computer programming and business/accounting.

In addition to producing financial statements, computers could be useful to small business in the following areas:

- monitoring status of purchase orders and accounts payable;
- control of inventories and reorder points;
- data on suppliers, parts stocked, time to process orders;
- sales and expense analysis;
- customer-service data;
- scheduling workloads;
- on-job training aids;
- formatted sales slips, purchase orders, etc.;
- text editing.

The last point cannot be overemphasized. Have you ever seen a secretary, after having carefully typed a long, important letter, come out of the boss's office tearing her hair out because he decided to change a word or two—and he wanted the letter out an hour ago? Conversely, have you been the recipient of a letter that is marred with correcting fluid? Besides, who would want to receive a letter typed by a dot matrix printer? I suspect that a Selectric-style printer with a business system that incorporates text editing would be a useful selling feature to small businesses (fringe benefits again). Unless I've missed the boat, or misread the fine print, such items just don't come with existing micro systems. Oh yes, while on the subject of text editing—remember Bill McLaughlin's article in *Kilobaud* No. 12, "ALL CAPS"?

A comment recently made in *Kilobaud* that small businesses can afford to spend $11,000-$12,000 for a microprocessor system may be true; but I would further modify this by saying:

1. It must be a system—hardware and software. The software portion must be tailored to meet the specific needs of the owner, and the ability to recognize and incorporate fringe benefits to suit the owner may be a key selling point. I estimate my time is worth $2000-$4000 to do an adequate analysis and related programming.

2. The system must be reliable in operation and aesthetic in appearance. If a system is down, it must be brought up again in minimal time—no rewiring circuit boards to interface components that didn't match originally. Such bargains are fine for hobbyists, but have little place in the selling of a business system.

3. Other costs such as service contracts, staff training and program debugging must also be considered in the sale of such systems.

After all considerations are summed up—and the above points only scratch the surface—perhaps $5000-$7000 is left for expenditure on hardware.

I do not intend to delve more deeply into the whys and wherefores mentioned above because many supporting points have been made directly or indirectly in past issues of *Kilobaud*. To summarize—I'm still looking for a *good* system that meets the forementioned requirements, that can be used to develop my programming expertise and that will serve as a model to sell to business. Until that system appears, I will probably compromise on a system to gain the necessary programming experience and, I hope, resell it to a new hobbyist "convert" at some future date. Meanwhile, until such systems are produced, my short term forecast is that Radio Shack's TRS-80 is going to cause hard times ahead for outlets that are geared mainly to the hobbyist market.

**Ted King**
**Slemon Park PEI**
**Canada**

*I recently heard a one-minute radio spot for IBM that expounded on IBM's small-computer systems for small businesses. There aren't many companies that can afford a nationwide advertising campaign such as that (if it is, in fact, nationwide). You can bet that as time goes by, the salesmanship demonstrated by the computer stores across the country (and in Canada) will be the determining factor between success and failure. Star Trek is fine . . . in its place!—John.*

### Articles on Network Communications

On page 17 of your January issue, you have an ad for a communications adapter. I would like to see some articles evaluating products such as this one. Also, articles on acoustical couplers and modems would be appreciated.

I believe data communications is an up-and-coming part of data processing the more I talk to microcomputer owners—they all have some plan to put their systems to productive uses, rather than just using them as toys.

I would like to see a questionnaire asking what percentage of owners' system design was dedicated to play, and what percentage to production. If sufficient time were spent designing it, such a questionnaire could be broken down further.

**Paul Krammin**
**Santa Rosa CA 95402**

*We have some good material coming up on this exciting subject, Paul. One of the earliest will be a review on such a communications adapter (by Russell Adams). —John.*

## PUBLISHER'S REMARKS

*(from page 4)*

in, say, *Kilobaud*. The article would reach at least 100 times more people . . . would result in a lot more prestige. Not only that, but *Kilobaud* PAYS for all articles, and pays very well.

Unless the person running the show intends to publish the paper for his own personal profit, there should be no objection to having a paper submitted for a show and also having it submitted for possible publication in a magazine.

With the average article pulling between $100 and $300, authors of papers are making quite a donation to computer shows when they give their hard work free of charge; it's the same as donating the cash. Many computerists would like to add a little extra memory or an I/O board to their system . . . instead they give away the money that could buy them. A recently published book of donated papers ran to over 300 pages . . . amounting to about $12,000 in donations from the authors of the papers. The book sold for $12, thereby bringing a very handsome profit to the publisher, all at the expense of the authors. Why should so many people spend all that time and effort just to help make one person wealthy?

### Reward!

Most businesses have a problem with employees wasting money on phone calls. Some make personal calls at the company's expense; some pick up the phone for any minor problem, where a short letter or note by mail would suffice; some make legitimate calls, but don't know how to stop talking. A microcomputer system can help with this situation.

We need a board to plug into the S-100 bus that will check all of the phone lines and record the numbers called, the date and time of the call, the length of the call, etc. It would also be helpful if the system could record the extension of the calling phone and perhaps a customer number. With a lookup table of toll rates vs time to different areas, the system could even make a fairly good estimate of the cost of each call. You might also build in a lookup table of customers vs phone numbers.

All this information would then be printed out either in real time or at the end of each day for a record.

The electrical end of this shouldn't be too difficult . . . but the program to put it together might take a while. As an impetus —if you are interested in developing such a system, I have an outfit that will put down $2000 for the prototype board and operating system . . . plus 5 percent royalty on the sales. Since just about every business that buys a microcomputer would probably want to add this board and system to its computer, the sales could be substantial.

Time is important. You might come up with a fantastic system in eight months, but the lesser system, already on the market in six months, could kill you. If you're going to try this one, get cracking.

## EDITOR'S REMARKS

*(from page 5)*

owners have the closest thing to a slick magazine (all your own) that I've ever seen. Harold Zallen will be publishing the *ICCD Journal* four times a year at a subscription cost of $18. (Kind of steep . . . but it has some good material and is very professionally prepared.) ICCD, PO Drawer 2790, Norman OK 73070.

*Robot Builders.* If you're even a little bit interested in robotics, then by all means drop a line to Michael Westvig, 208 Via Colorin, Palos Verdes Estates CA 90274. Send him an SASE for more information.

*IMP-16 Owners, unite!* Frederick R. Holmes, 101 Brookbend Ct., Mauldin SC 29662, is publishing a newsletter for you folks interested in home-brewing IMP-16 systems. (In addition, you should get a subscription to National Semiconductor's *Compute.* Compute/208, National Semiconductor, 2900 Semiconductor Dr., Santa Clara CA 95051.)

*Micro.* Now *all* of you 6502 owners can band together in a grand conspiracy . . . through the pages of *Micro.* Robert Tripp (who formerly put out *The Computerist*) has a great semi-magazine going here, directed toward all 6502 owners (Apple, KIM, OSI, PET, Jolt, Data Handler and more). The cost is $6 per year (6 issues) and I think you'll find it worthwhile. Micro, 8 Fourth Lane, So. Chelmsford MA 01824. (And . . . if you're among the 6502 group, you should certainly be getting the "KIM-1/6502 User Notes" from Eric C. Rehnke, 109 Centre Avenue, W. Norriton PA 19401. $5 for six issues . . . and check into getting the back issues!)

*The Computer Hobbyist.* It's alive and well! Bill McLaughlin may not know how to spell hobbyist (hobbyiest), but he sure puts out a neat newsletter! About all I can say is that it covers a wide range of topics and should be of interest to just about anyone (it's subtitled "The 2650 Computer User Notes" but really has a lot of general-purpose information . . . . and I hope he keeps it that way). I couldn't find the price! (The Computer Hobbi[e]st, Box 158, San Luis Rey CA 92068.)

# BOOKS

*(from page 13)*

circuitry, including equations for setting the circuits to the time and frequency that the reader requires.

*IC Timer Cookbook*'s appendixes include manufacturers' data sheets for the devices covered in the book and a second-source guide. Finally, the book contains a "Bibliography of IC Timer Design Ideas." Basically, this is a list of articles from various professional electronic magazines that have covered IC timers. This is a nice feature if you have access to the magazines listed.

As in his other cookbook, Jung has included references to useful data sheets and application notes for the reader who requires more information on certain ideas. You have only to look through this book to see what an important part the IC timer circuitry plays in today's electronic circuitry.

I suggest that you take a look at this book and see if it would make a worthwhile addition to your book collection. It is a good selection for both those who use IC timers and those who want to.

**Michael Black**
**Montreal Quebec**
**Canada**

# LEGAL/BUSINESS FORUM

*(from page 7)*

concepts. It is a mandatory requirement that copies be deposited with the Library of Congress within three months after publication. The Copyright Law provides for fines if the deposit of two complete copies is not made within such time. However, exceptions can be made for material the Library of Congress neither needs nor wants.

Registration with the Register of Copyrights is not mandatory. However, such registration is a prerequisite to bringing a lawsuit for the copyright infringement. The Register is free to allow or require the deposit of printouts of computer programs rather than a tape or disk.

It is interesting to note that although the Register of Copyrights has been accepting computer programs for registration for over ten years, only about 1300 programs have been registered. It appears that proprietors of software do not wish to open their programs up to the possibility of infringement by the public, which has access to everything registered with the Register of Copyrights.

## Infringement and Remedies

Anyone who violates any of the exclusive rights of a copyright owner is an infringer of the copyright. Those exclusive rights have been discussed previously. The owner is entitled to get an injunction prohibiting the offender from further infringments, such as distributing infringing copies. The owner is also entitled to damages equal to his lost profits plus the profits of the infringer. The latter prevents the infringer from making money by virtue of his wrongdoing.

In lieu of such actual damages, the owner of the copyright can elect to receive statutory damages, which vary from $250 to $10,000 as the court considers just. If the court finds the infringement was willful, the ceiling on statutory damages expands to $50,000. If the court finds that the infringer had no reason to believe his act was an infringement, the statutory damages can be reduced to as little as $100. In any event, the copyright owner receives his court costs and attorney's fees from the infringer.

The Copyright Act also provides some criminal penalties for willful infringement of copyright for the purpose of commercial advantage or private financial gain. There are also criminal penalties for giving fraudulent copyright notices, for removing a copyright notice or for making a false representation with respect to registering a copyright claim.

Where there is a willful infringement, the infringer is subject to having all property used for the making of infringing copies seized and forfeited to the United States.

## Fair Use

There are some exceptions to the rights of a proprietor of a copyright. One exception is embodied in the Doctrine of Fair Use. Fair use defies precise definition. However, broadly speaking, it means that a reasonable portion of a copyrighted work may be reproduced without permission of the author for a legitimate purpose that is not competitive with the copyright owner's market for his work. This doctrine most often arises when a teacher copies copyrighted material for distribution to students.

The courts have generally stated that whether a use constitutes a fair use must be decided on a case-by-case basis. The criteria used by the courts for determining whether a use is fair are: (1) the purpose and character of the use; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; (4) the effects of the use upon the potential market for, or value of, the copyrighted work.

Those factors led me to the following observations. If you plan to take very short routines and subroutines from previously copyrighted materials and plan to use them to create a program for your own computer, I tend to think that would be a fair use. That assumes the purpose is for your own use only and that you take a relatively small portion from each of the previously copyrighted works. If, on the other hand, you merely make patches between existing routines to make them compatible with your system, you have created a derivative work, which would infringe on the rights of the owners of the programs used.

The bottom line, it appears, is that you, as a hobbyist, will have to do a considerable amount of "reinventing the wheel" unless you are prepared to purchase previously copyrighted material which may then be adapted to any individual system. However, that adaption is a derivative work and cannot be sold by the adapter to other hobbyists with the same system unless permission is received from the owners of all copyrighted materials from which the adaptations are made. Of course, the adapter is free to sell his patches. Then other hobbyists can purchase authorized copies of the underlying material and incorporate the patches.

It should also be noted that if a programmer sat down and wrote an operating system for his computer without the use of any routines or subroutines appearing in previously copyrighted materials, and if the end product were a verbatim copy of previously copyrighted materials, there would be no infringement. The Copyright Law only prohibits copying, it does not protect against the independent creation of an identical work.

The workings of copyright are sure to frustrate the hobbyist who tries to maximize utility while minimizing costs. But the purpose of copyright is to give economic benefit and protection to authors. It makes them very happy to know that others cannot legally appropriate the fruits of their labor.

In this column I have tried to give you, very briefly, an overview of law of copyright. You should be aware that there is much more omitted from this discussion than has been included. Don't be your own lawyer based on this column. Copyright is a complicated area of the law and certainly can't be adequately discussed in a few pages. If you have or think you have a copyright problem, see an attorney.

# Kilobaud's Mystery Progam

## *are you ready for this?*

*Tom Rugg*
*Phil Feldman*
*PO Box 2485*
*Los Angeles CA 90024*

We bet you thought it would never stop. Over and over again the same scene has been repeated: Your new issue of *Kilobaud* arrives and you begin to slowly make your way through it. What kinds of things will you find this month?

As you flip the pages, you find the same old thing—article after article filled with well-written, concise, valuable information; clear explanations of every conceivable aspect of microcomputing; listings of fun and useful programs, complete with easily understandable descriptions of what they do and how they work.

Is this monotony ever going to end? Will *Kilobaud* ever change its policy and decide to publish something that can't be understood? Yes and yes, in that order.

This month, at long last, *Kilobaud* is departing from its practice of providing you with useful and understandable material about the world of small computers. Instead, you're getting the first (and possibly the last) *Kilobaud Mystery Program!*

**The Program**

What does the program do? We're not going to *tell* you; but we're glad you asked. There are two ways you can satisfy your curiosity, which must certainly be more than you can bear by now.

First, you can look at the Mystery Program listing and try to "walk through" it to figure out what it will do. This might appeal to those of you who are incurable problem-solvers or masochists. Needless to say, some efforts have been made to disguise what it does.

The second approach will give you the answer more quickly. Find a handy nearby computer and run the program. Make sure you don't make any mistakes when copying the program, of course. In particular, don't forget to include those semicolons at the end of some lines.

You'll discover that the program is even interactive! When you run it, it will give you instructions on what to do next. Follow the instructions and then run it again. Amaze your friends! Amaze yourself! Be the first on your block! Be the first off your rocker!

**Compatibility Notes**

The program was written in Altair 8K BASIC and fits in an 8K machine (along with BASIC itself). We didn't use any fancy nonstandard techniques, so you should be able to run the program using other versions of BASIC, too.

The only problem might be the use of the CHR$ function in lines 800 and 900. The CHR$ function provides the ASCII equivalent of a decimal number. So, line 900 prints the ASCII character that corresponds with the decimal value in the variable T. If T happens to be 65 at the time, an A is printed.

If your version of BASIC doesn't have the CHR$ function, you'll have to substitute either an equivalent statement or an equivalent subroutine to accomplish the same thing that CHR$ does at these two places in the program. Most versions of microcomputer BASIC that we know of have either CHR$ or a direct replacement for it, so this shouldn't be a problem for you.

Well, what are you waiting for? Go find a computer! Who knows what lurks in the mind of *Kilobaud*?■

```
110 REM: **** A KILOBAUD MYSTERY PROGRAM ****
120 REM: FROM THE APRIL, 1978, ISSUE.
130 REM: BY TOM RUGG AND PHIL FELDMAN
150 DIM A(42)
160 FOR J=3 TO 28
170 A(J)=J+62
180 NEXT J
190 FOR J=1 TO 10
200 A(J+29)=J+47
210 NEXT J
220 FOR J=1 TO 2
230 A(J)=J+31
240 NEXT J
250 REM: NOW IT GETS MORE MYSTERIOUS
260 A(A(1)-3)=A(A(2))-7
270 T=0
280 FOR J=1 TO 9
290 T=T+J
300 NEXT J
310 A(T-5)=T+1
320 A(A(T-T+1)+T/5)=T+INT(T/3)+3
330 T=T-A(2)
340 J=T/4
350 A(T*J+2*J)=J*T+(T/J)-1
410 GOSUB 500
500 GOSUB 700
510 READ T
520 IF T<>0 THEN 550
530 GOSUB 700
540 GOTO 510
550 IF T<0 THEN 580
560 GOSUB 800
570 GOTO 510
580 IF T=-1 THEN 1230
590 J=ABS(T)
600 READ T
610 FOR K=1 TO J
620 GOSUB 800
630 NEXT K
640 GOTO 510
700 T=A(24)-A(29)
710 PRINT
720 FOR J=1 TO T/2
730 GOSUB 900
740 NEXT J
750 PRINT
760 RETURN
800 PRINT CHR$(A(T));
810 RETURN
900 PRINT CHR$(T);
910 RETURN
920 DATA -5,1,3,18,20,11,14,1,8,-2,17,14
930 DATA 2,0,22,20,27,1,6,7,14,7,22,11,16,9,1,14,11,16,7,21
940 DATA 1,39,33,30,1,3,16,6,1,39,34,30,-4,40,-1
950 DATA 1,3,9,3,11,16,2,0,16,17,25,1,20,7,15,17,24,7,1
960 DATA 39,32,30,29,1,39,35,30,29,1,3,16,6,1,39,36,30,-1
970 DATA 20,11,6,6,14,7,-4,40,1,25,10,3,22,42,21,1,3,1,22,10,20
980 DATA -2,7,1,22,17,7,6,1,21,14,17,22,10,41,0,-4,40,6,7,14,7
990 DATA 22,7,1,39,37,30,29,1,39,38,30,29,1,3,16,6,1,-2,39,30,-1
1000 DATA -3,1,10,3,2,1,10,3,2,1,10,3,2,1,10,3,2,1,0,11,42,15,1
1010 DATA 3,1,5,17,15,18,23,22,7,20,29,1,16,17,22,1,3,1,28,-2
1020 DATA 17,14,17,9,11,21,22,2,0,4,27,7,-1
1230 PRINT
1240 END
OK
```

*The Mystery Program.*

*George Young*
*Sierra Union High School*
*Tollhouse CA 93667*

*Bob Grater*
*Lockheed Space Systems*
*Sunnyvale CA 94086*

# Make Your Own PC Boards

## *start with a universal wire-wrap board*

*Hey, this universal wire-wrap board looks like something I could use. I wonder if anyone will ever tell me how to get it off the published page and onto a circuit board?*

We hear you! How many times have you seen a circuit-board layout presented with an article, only to find that there is no source for the board? All you have is that layout on the page.

The average ham or computer phreak would make his own circuit boards if: 1. He knew how. 2. It didn't cost a small fortune. 3. He didn't need a lot of special equipment.

In the early 1977 issues of *73 Magazine,* there were several excellent articles on making your own circuit boards. The average hobbyist does not want to go into PC board production; he only wants to make a single board for his own use, or maybe one for a buddy. He may want to make boards utilizing the schematics from several different articles and be willing to sacrifice perfection for low cost and availability. He can't, and won't, spend more money for the materials to construct circuit boards than he would have to spend for all the individual boards he needs.

### The Artwork

The preliminary circuit-board layout complementing an article is called the *artwork.* This is the circuit wiring and the solder attachment points, called *pads,* for all the ICs, transistors, resistors, capacitors, etc. Usually the artwork is included with the article, as in this case. If no artwork is published with the article, you might assume that the author did not use a PC board to construct his device. He may have used wire wrap, perfboard, or some other technique.

Now, whenever artwork is published, we want to be able to use it. Therefore, we need a method to lift the artwork off the printed page so we can make our own circuit board. However, some precautions are in order. Some interesting things can happen in the process of getting artwork into print. Artwork that is specified as full scale (1X or 1:1) may not turn out that way. For example, a circuit board that I wanted to use recently was reproduced in the magazine at one-half scale. It actually turned out to be 7/16 scale. To verify the scale, lay an IC (14, 16 or 24-pin) over the 1:1 artwork. If it is full scale, fine; if not, we need the capability of lifting it off the page *and* reproducing it to full scale.

### Doing Your Own Artwork

The least expensive method of doing your own artwork is to use graph paper, ruled ten lines to the inch, available at most stationery stores. Using graph paper, you can lay out your artwork in 1X, 2X or 4X.

If you are going to follow the PC board reproduction scheme outlined in Kilobaud Klassroom No. 4 (September 1977), use a 1X layout. This method can be used to lift any article from the printed page, but I recommend it only for the simpler circuit boards. It has been used to make the circuit boards for the TVT-6L, designed by Don Lancaster, and can be used for complex boards, but not easily. If you plan to use the layout for photographic reproduction of the circuit boards, do your artwork in 2X or 4X. Then, when the artwork is reduced to 1X, all the dimensional errors are reduced proportionally (for example, 2X to 1X will reduce the errors by one half).

You can use the artwork by going directly from the graph paper to the photographic film. However, you'll need a green filter over the camera lens to eliminate the green lines from the graph paper. I prefer to do the entire layout on the graph paper in pencil so I can make changes with an eraser. Then I use a light table (a frosted sheet of glass or plastic with a light source behind it) with a sheet of white paper placed over the graph-paper layout to produce a high-contrast black-and-white drawing. A felt-tip pen can be used for the ink work, but india ink is better. Either will produce a good-quality drawing.

Templates for the pads, ICs, etc., can be commercial or homemade. Commercial templates in 1X or 2X are available from Tangent Templates, Box 20704, San Diego CA 92105. My homemade template consists of a series of holes drilled in 1/8 inch Plexiglas, but any thin plastic can be used. The graph paper provides the locations of the pads.

The whole idea is to sacrifice a little on precision and hold the cost down. So far, we have had to buy graph paper, india ink (get Pelikan), and perhaps a pen to apply it with. If you have to buy an india ink pen, get a Rapidograph. It'll cost like crazy, but it's a lifetime investment. A #2 tip is a good place to start.

You can, of course, spend a lot more for material to get the work done easier, faster or more precisely. These are trade-offs that each individual must make for himself.

### Using the Published Artwork

Recently I wanted to use some artwork published with a magazine article. For years I had been transferring the published artwork to the circuit board by hand. The time had finally come when the circuit I needed was too complex for this procedure, so I had to teach myself how to transfer the artwork from the printed page to the circuit board. There are three methods that Bob and I have found successful.

The first method was covered in Kilobaud Klassroom No. 4. Simply place your copper clad under the published 1:1 artwork, punch the pad pattern onto the copper clad and reproduce the original artwork by hand.

Bob uncovered the second method in *CQ Magazine* (May 1977, p. 46). You can lift the published artwork off the printed page using Thermofax equipment and overhead-projection Thermofax film to obtain a black-and-white transparency of the artwork.

The same process is now available for use with Xerox equipment. A positive transparency can be made using their 3R459 Transparency Film. This material costs over $30 per box of 100 sheets.

The black-and-white film can then be used to transfer the artwork directly to copper clad sensitized with a positive-acting photoresist. However, most presensitized copper clad is sensitized with a negative-acting photoresist, and most readily available sensitizing material comes in the negative-acting form. The black-and-white Thermofax or Xerox positive can be reversed by contact printing to produce a photo negative for use with negative-acting photoresist. (More on the contact printing process shortly.)

If the artwork is 1X, if a Thermofax or Xerox machine is handy, and if you can get the required copper clad with positive-acting photoresist, this is an easier method of getting the artwork lifted off the printed page. Otherwise, you will need the following method.

**Using Photography to Lift Artwork**

Artwork can be lifted off the printed page using a graphic-arts camera, a special camera used in the printing industry. The artwork is usually removed physically from the magazine and placed on an easel, vacuum frame or copyholder. It is then shot with the graphic-arts camera directly onto high-contrast film to make a photographic negative of the original. This is a one-step process for reproducing the photo negative and is the fastest way to get the job done. It is also the most difficult method for us to use since we have to locate one of these special cameras. Many printers and some high schools and junior colleges have them, but most of us will have to use the process described below.


*Raw-material source for safelight/oven and developing tray.*

A camera and enlarger can be used to bypass the graphic-arts camera. In fact, we can even get by without the camera. We will need access to an enlarger. To lift any reasonably complex artwork off the printed page, some photographic process must be used. Here is a practical way to do it. If you have the necessary photographic skills, proceed; if not, get someone to help.

The artwork is first shot with a photocopy setup such as an enlarger with the lamp housing removed and a sheet of ground glass replacing the negative carrier (tracing paper can also be used in place of the ground glass).

The artwork is placed on the easel and sharply focused on the ground glass (which is then replaced with a sheet of film) and a shot is made. The processed film is a photo negative that will fit our enlarger's negative carrier. High-contrast lithographic film (available from offset printing suppliers) can be used, but any good-quality film will work.

After the negative has been developed, fixed, washed and dried, we can proceed to the next step. The enlarger is reassembled to its normal configuration. The artwork negative is placed in the enlarger's film carrier and focused on the easel to obtain a 1X image. To insure a 1:1 scale, place an IC over the projected image and adjust the image size until the pads in the image exactly fit the IC pin spacing. Now we can make a print.

Enlargements must be made on lithographic film. It is expensive, but no more so than any film purchased in sheet form. If you go into partnership with a friend you can cut the cost.

Next, a 1:1 print is made from the negative in the enlarger. Lith film should be processed in special developer, but any developer used for photo enlargements will yield satisfactory results. The processed print is a photographic positive of the original artwork at the correct scale. We have now reached the point we would be at had we used the Thermofax method.

The positive is now contact printed on lith film to produce the final 1X photographic negative of the original artwork.

Note that we can change the scale of the original artwork in this process, but using the Thermofax method, we were stuck if the original scale was not 1:1. Also note that the line positive, or high-contrast positive, produced in the camera-enlarger method might at first seem wasted. It is only produced in order to get from the negative to the 1X reproduction. However, far from being a useless by-product, it can be used to make as many negatives as you want so that your friends can have a copy of your PC board layout. Now that we have the photo negative, we can proceed to make the circuit board.

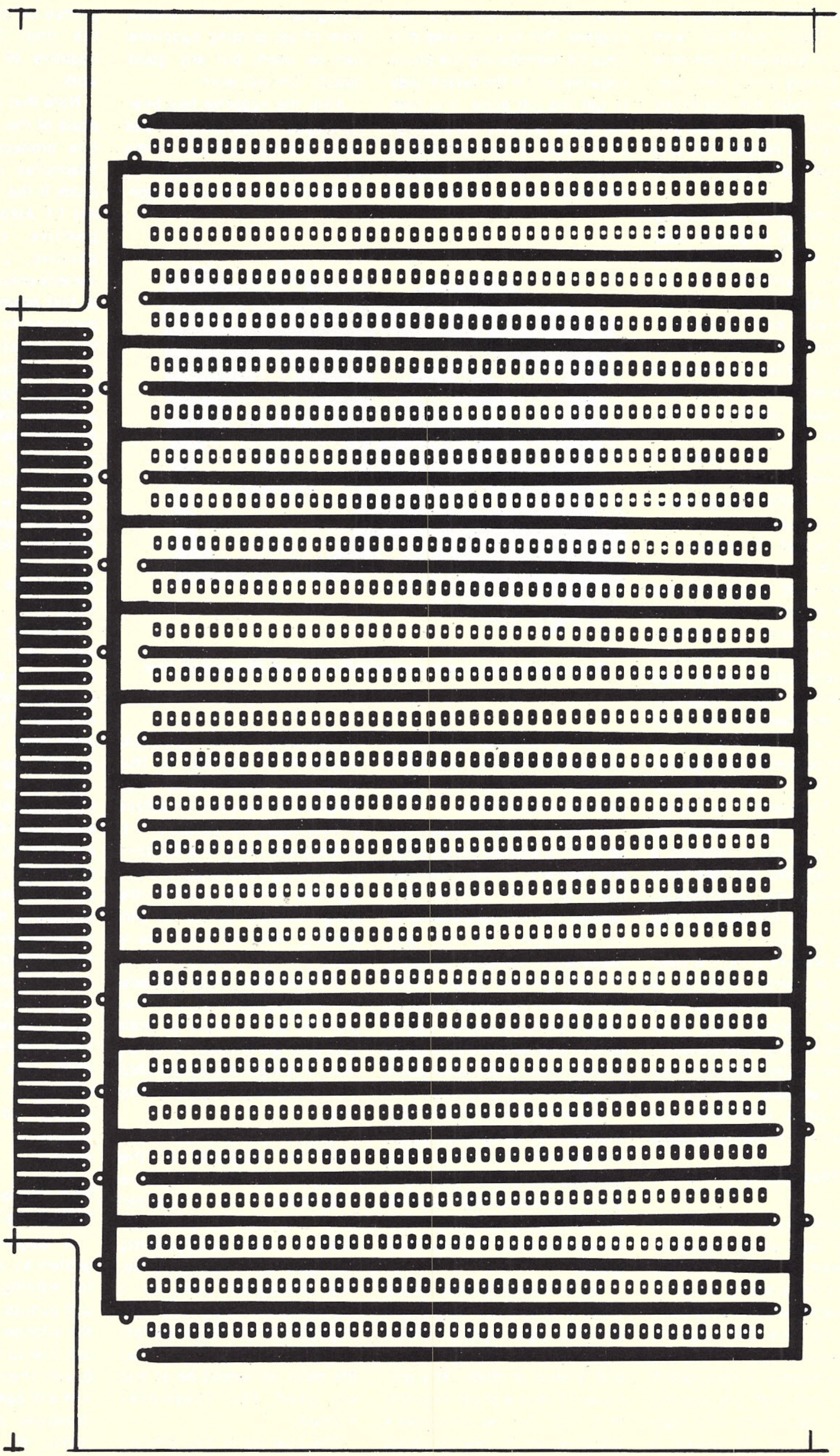
**Making the Photographic Circuit Board**

The first step was the artwork—your own, or that published with an article. The second step was to get a photo negative of the artwork at a 1:1 scale. Now, we have to transfer it to the copper clad board.

Suntronix Company advertised a PC kit in *Kilobaud* No. 2 (February 1977) for $14.95 plus shipping. It contains several pieces of copper clad (single and double-sided), about two pounds of dry etch and a pint of immersion tinplate solution. You will have to buy some copper clad and etch. You don't have to tin-plate your boards, but it makes them look a lot more professional. For the etch, you can use ferric chloride, ammonium persulfate or cupric chloride. The Suntronix kit includes persulfate.

**The Safelight/Oven and Developing Tray**

A Buglite has been suggested as a suitable safelight for working on sensitized circuit boards. After a few trips to the kitchen with all the lights out, first to prebake and dry the board, then to dry the resist, and still again to evaporate the developer and post-bake the resist, an alternative to the



*Artwork for the universal wire-wrap board. Scale is 1:1.*

kitchen oven was suggested (insisted upon) by my wife.

The 60 Watt Buglite was mounted in an empty solvent can, and now serves double duty. The light produces safelight illumination, and the trapped heat is just about right for all the heating processes required in working on the boards. The "oven" is simply the upper surface of the can.

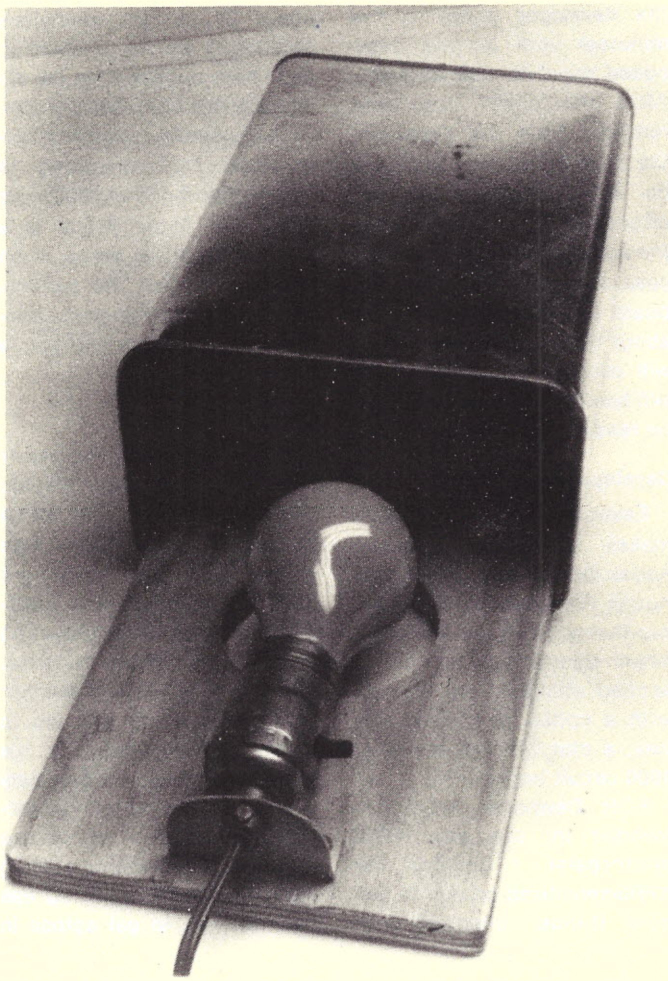
Cut out one side of a second can and fold the edges back to eliminate the sharp surfaces. This will become the developing tray. Any metal tray can be used, but this one is cheap and easy to make. To return unused developer to its storage container, just unscrew the cap and pour the solution without spillage through the opening.

*Don't use a metal tray for the etching process.* The etch will eat up the tray. Use either a glass or plastic container—or the milk carton "boat" suggested in Kilobaud Klassroom.

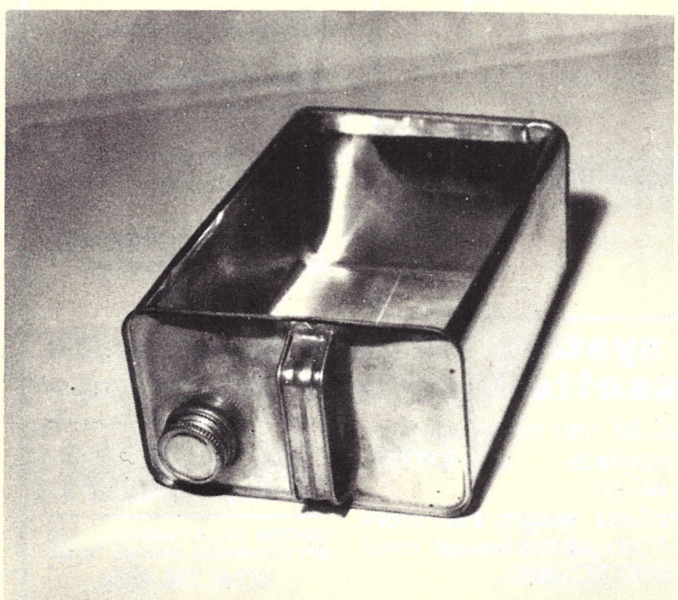
Presensitized circuit board costs about four times as much as sensitizing your own. Sensitizing material is available in spray cans from General Cement (catalog no. 22-230 or 22-231).

Yours truly, George Young, followed the directions on the can, and half a can later I still did not have a decent sensitized board. I tried everything I could think of, and then the spray nozzle on the can plugged up. After uttering a few suitable expletives for this typical Murphy's Law situation, I removed the spray head, cleaned it out, and apparently in the process enlarged the hole in the nozzle. A blob of resist hit the board. More expletives. I cleaned up the blob with a cotton pad, and suddenly realized the board looked pretty good in the yellow light. I wiped the whole board, and it still looked good. It wasn't very smooth, but it was a lot better than the results I had been getting. I dropped it on the Buglite oven and the heat caused the resist to flow out over the surface. In a few minutes it was dry. I deliberately turned on the room lights, exposing the board. It looked very good.

I removed the resist once more, cleaned and dried the board, and when it had cooled, repeated the process. I just took the blob produced by the spray can, wiped it over the surface with the cotton pad, and dropped it on the oven. Again it looked good. I burned and etched the board, and had my first photographic circuit board. (The process of exposing the sensitized circuit board through the negative is called *burning* in the printing industry.) In the process, I had



*The safelight/oven.*



*Developing tray.*

learned a method of getting the relatively inexpensive photoresist onto the board using far less resist than the spray technique. I've since learned that other resists can be applied in the same manner with equally good results. The cotton pads are those used in the offset printing industry, and are available from your local AB Dick supplier (listed in the Yellow Pages). Ordinary cotton from the household first aid cabinet should also prove satisfactory.

**Exposing the Sensitized Copper Clad**

The least expensive exposure unit is Old Sol. Place the line negative in contact with the sensitized copper clad working under the safelight in a darkened room. Make sure the image you are going to create is rightside up. For you photographers, emulsion faces emulsion; for nonphotographers, you'll probably have to ruin the job at least once to find out which surfaces face each other. Remember that you want the finished product to look the same as the original artwork.

Place the negative and the sensitized board between two sheets of glass or a sheet of glass (with the glass on the negative side) and a piece of cardboard or other stiff backing. Tape the sandwich together, being careful that the tape does not obstruct the surface to be exposed. The "sandwich" is then exposed to bright sunlight for at least five minutes. Ten to twenty minutes won't hurt—it is difficult to overexpose the resist.

After the sensitized board is burned (exposed), it must be developed. Return to the darkened room, and, working under the safelight, place the board in a metal tray and slosh very gently back and forth in the developer (three to five minutes) until all the unexposed resist is washed away. After the board has been removed from the developer, the room lights may be turned on. All photoresists are soft after development. *Do not wipe the board off.* Drain off any excess developer and place the board on the Buglite oven. In about 15 minutes the developer will evaporate and the resist will harden. The board will then be ready for the etch.

### Developers and Resists

Eastman Kodak Company makes several photoresists. Kodak Ortho Resist (KOR) and Kodak Photo Resist (KPR) are two that you can use. KPR3 is a newer Eastman resist that is applied straight from the can with a cotton pad. Used this way, a pint can will sensitize 1000 circuit boards.

KOR Developer is recommended for use with KPR3 photoresist. I develop my KPR3-sensitized boards in lacquer thinner charged with about 20 percent methylene chloride. I am not a chemist, and I have no idea what I'm doing with this stuff, but it works.

The Eastman resists are fairly expensive—something in the neighborhood of $20-25 per quart. GC Spray Resist is a good alternative; spray it on and wipe it over the surface. When the nozzle clogs up, convert it to a blobber and you're in business.

### Other Assistance

Now, here is another possibility for photographic assistance that you may not be aware of: Almost every high school has the equipment (camera, enlarger and darkroom) and the labor (students) to make your photo negative. And every high-school teacher can justify doing this job in terms of education for his students. In fact, it is precisely what he is looking for—a practical application of course material. It represents putting into practice the skills he has been striving to get across in the classroom. Far from imposing on the teacher, you will be doing him a favor. The student will get an A for your photo negative, you will get a job done and the teacher will chalk up another plus for achievement.

If the job is done as an educational process, some ground rules need to be spelled out. What can be done commercially in a few hours may take several days in the educational world. Learning takes time, and if you utilize the facilities that you, as a taxpayer, own, please make allowance for this fact.

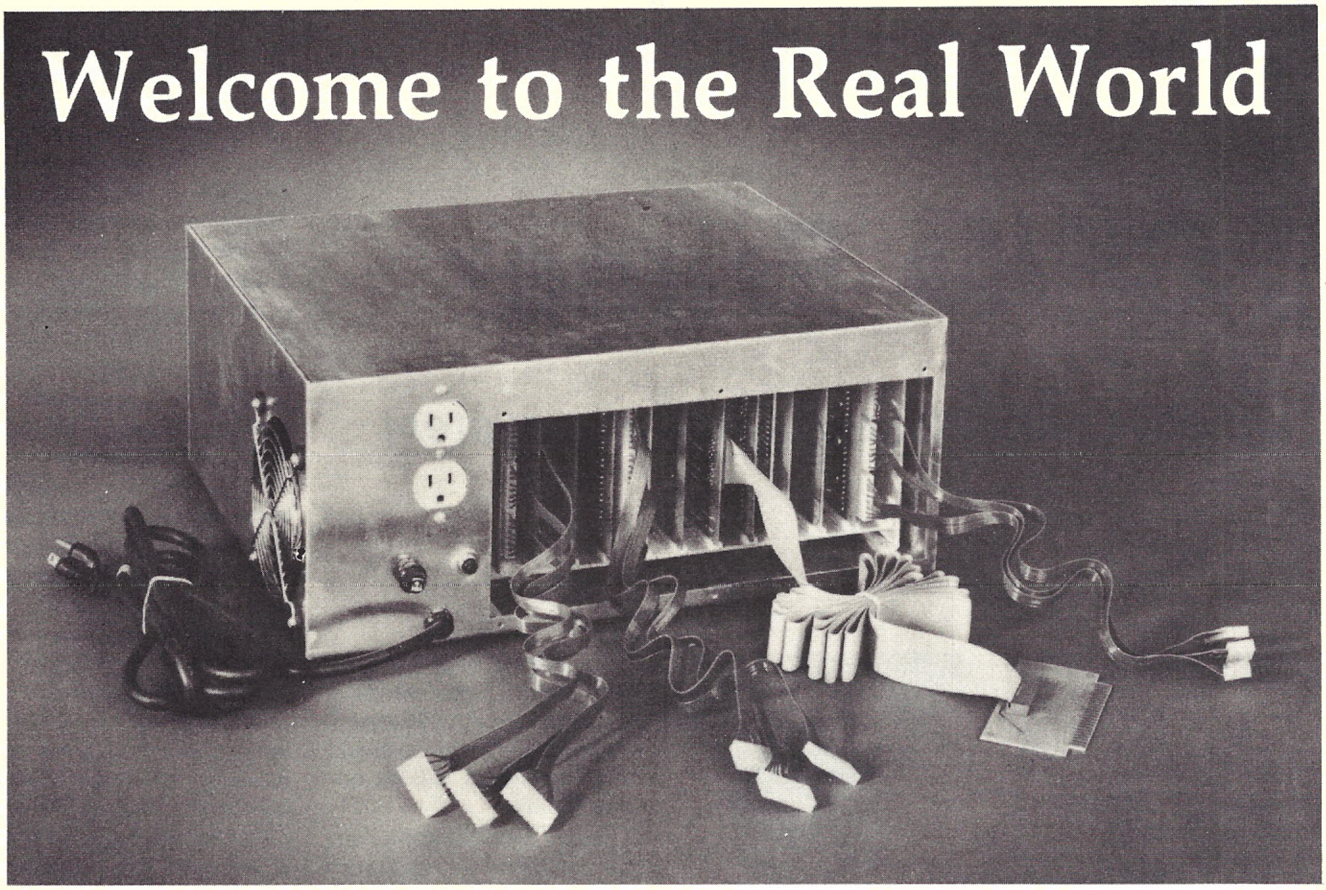
If your local school is not equipped for the job, drop me a line and I'll try to help. Send $2 for each negative you want reproduced, your artwork and any other pertinent details. That should more than cover the cost of materials. My students are always looking for fund-raising activities, and if I can combine that with their education, so much the better. If I get too many requests, you might have to wait too long for the kids to do your job. If it looks like this is going to happen, I'll send your stuff back with your money and advise you accordingly. If you plan to mail the artwork, scale it in 1X and *don't fold it.*

### Summary

This started out as an article on a universal wire-wrap board that could add 4K of memory to your machine. We ended up with an attempt at giving you the capability to produce your own circuit boards at a cost commensurate with other aspects of our hobby.

Photographic circuit boards can be made by the average home-brew enthusiast without placing undue strain on his pocketbook. We have tried to give you a method of getting artwork off the printed page and onto your copper clad so you can use all the work others are making available to us.

Need 4K or 8K of memory for your computer? Print it and wrap it up! ■




# with the Real-World Interface from The Digital Group

A computer should have a purpose. Or as many purposes as you can imagine. Because a computer belongs in the real world.

And now, the Digital Group introduces the Real-World Interface. A system component that's actually a system in itself, and specifically designed to help you get your computer to control all those tasks you know a computer can control so well.

Automate your sprinkler system. Heat and cool your home. Guard against burglars. Shut off lights . . . It's all a part of the Real World, easily controlled with the Digital Group Real-World Interface.

Our Real-World Interface is initially made up of three basic components — motherboard and power supply, parallel CPU interface and cabinet — plus three types of plug-ins: AC controller, DC controller and prototyping card. The recommended software packages are Convers, Assembler or Maxi-Basic, in that order.

**Some of the features include:**

**Motherboard & Power Supply**
- 12 slots — 11 control cards, one for the interface card
- +5V DC±5% @ 1A, +12V DC ±5% @ 1A, −12V DC ±5% @ 1A contained on board
- May be free-standing (with care)

**Parallel CPU Interface**
- All buffering for Data Out (25 TTL loads), Address (25 TTL loads) and Data In (10 TTL loads)
- Includes cable and paddlecard for connection to dual 22 on Digital Group CPU back panel. Two 22-pin edge connectors included
- Requires two output ports and one input port

**AC Controller**
- Eight output devices (2N6342A-2N6343A, −12 amp Triacs); Each output 240V AC max, 12A max RMS
- Control AC motors, lamps, switches, etc.
- Opto-isolated (MCS-2400 or equivalent)

**DC Controller**
- Eight output devices (2N6055) each output up to 50V and up to 5A
- Control DC motors, switches, solenoids, etc.
- May use internal +12V DC for load or external DC up to 50V DC

**Price**
- For the motherboard and power supply, parallel CPU interface and cabinet, our kit price is only $199.50, or $260 assembled. Now *that*'s down to earth.

We've only just begun our Real-World Interface System. There are many more plug-ins and applications coming along soon. So write or call The Digital Group now for complete details.

And welcome to our world.

the digital group INC

**P.O. BOX 6528 DENVER, CO 80206 (303) 777-7133**

**D12**

*Dr. John F. Stewart*
*University of Miami*
*Box 248237*
*Coral Cables FL 33156*

# CP/M Primer

## *a most sophisticated operating system*

*The author works on an Imsai 8080 in the University of Miami's Hertz Computer Lab.*

Many fine articles have appeared in *Kilobaud* describing the principles of operating systems, but, as yet, no one has taken it on himself to present a detailed description of one of the several commercially available operating systems. In this article, we will take a look at the disk-based CP/M system written by Digital Research. In particular, we will look at the version of CP/M that is currently available on the Imsai 8080 microcomputer system. There are only minor differences between the Imsai version and the original, so most of the following will apply to both.

The CP/M operating system is currently available from Digital Research for $70, including documentation and system diskette. I estimate it would take three to six man-months for a sophisticated programmer to produce an operating system with CP/M capabilities. Thus, one of the original problems inherent in the microcomputer field — a lack of inexpensive software — has apparently been alleviated.

### Environment of CP/M

The essential structure of the CP/M operating system is shown in Fig. 1. The CCP is the Console Command Processor — the part of the operating system with which a user converses. A wide variety of commands is available, and these commands will be discussed below.

Basic Input/Output System (BIOS) is the section of CP/M that deals with input/output commands to all peripheral devices except the floppy disks. This includes I/O to Teletype, CRT, printer, etc. A nice feature of BIOS is that the system I/O routines are available to the user through appropriate subroutine calls in assembly-language programs. This capability constitutes a powerful addition to the assembly-language arsenal.

Basic Disk Operating System (BDOS) interfaces the system with the floppy-disk peripherals. Again, these routines are available to the user, eliminating what is typically one of the trickiest aspects of assembly-language programming — that of I/O programming.

The first $100_{16}$ ($256_{10}$) bytes of memory are used primarily as a scratchpad by the system. Various system parameters, such as where to jump on a restart, are contained in this area. In addition, a fair amout of it is available to the user. In particular, the default location of the top of the stack is location $FF_{16}$ ($255_{10}$).

Finally, the Transient Program Area (TPA) is the area of memory available for user programs. It comprises the bulk of memory, even in the 16K system where it is $26FF_{16}$ ($9983_{10}$ or 9.75K) bytes in length. In addition to user programs, all CCP transient commands are executed in the TPA. Thus, all user and most service programs originate at location $100_{16}$.

The CP/M system is a disk-based system, so that an important part of the environment confronting the user has to do with the way the diskettes are structured. The diskettes are composed logically of 77 concentric tracks numbered from outside to inside as track 0 through track 76. The first two tracks (0 and 1) are used to hold the CP/M system, which is bootstrapped into memory as indicated in Fig. 1, when a cold-start procedure is initiated. Tracks 2 through 76 are available for the directory (usually on track 2) and user or system disk files (programs or data files). Each track contains 26 sectors, each of which is capable of holding 128 bytes of information. Total disk capacity, then, is a little in excess of 250K bytes, of which just over 240K is available for user files.

There are several important points to make about the disk environment. First, it

| File Type | Meaning |
|---|---|
| BAS | BASIC program. |
| ASM | Assembly program. |
| SUB | Submit file. |
| INT | Intermediate BASIC. |
| PRN | Assembly results. |
| HEX | Assembly output. |
| COM | Command file. |

*Fig. 2. File types.*

is not necessary for the user to specify where on a diskette a particular file will go. The (BDOS) system automatically finds the necessary space and keeps a record of the name and location of each file in the diskette directory. This saves the user the trouble of remembering where a particular file is located. Names of disk files are made up of three parts:

1. The first letter is used to indicate which drive (A or B) the diskette is on. This letter is optional if the operating system is told to assume that all files are on a particular drive.
2. The second part of the name is called the file name. It consists of from one to eight letters and/or numbers.
3. The last part is the file type. File type is used to indicate whether a file is a BASIC program (BAS), an assembly-language program (ASM), etc. A list of file types is given in Fig. 2.

Valid disk file names are shown in Example 1.

A file name as defined above constitutes an unambiguous file reference. In many cases, it is desirable to refer to a whole set of files with similar characteristics. This is done through the use of an *ambiguous file reference*. File references can be ambiguous in one of two ways:

1. An asterisk can be used in place of either file name or file type to indicate any file name or file type. Thus, *.BAS refers to all BASIC language source files while MYPROG.* refers to all files named MYPROG, no matter what type they are.
2. One or more question marks can be used in place of characters in either file name or file type to indicate that any character in that position is acceptable. Thus, files TEST1.BAS, TEST2.BAS and TEST3.BAS could be referred to as TEST?.BAS.

```
A:MYPROG.BAS
B:F12.ASM
PIP.COM (uses default drive)
```

*Example 1.*

| |
|---|
| BIOS/BDOS $3100_{16} - 31FF_{16}$ |
| CCP $2800_{16} - 30FF_{16}$ |
| TPA $100_{16} - 27FF_{16}$ |
| System Area $0 - FF_{16}$ |

*Fig. 1. Structure of CP/M 16K system.*

**Console Command Processor**

As stated above, the CCP is the part of CP/M with which a user communicates. CCP prompts the user with a letter that indicates from which disk drive the system has been taken (also the default disk drive for file references) followed by a greater-than character (i.e., A> or B>).

Two types of commands are possible in CCP. There are built-in commands such as DIR (list directory of default disk), ERA (erase a file), REN (rename a file), TYPE (list a file) and SAVE. These commands are referred to as built in since the code for them is in the CCP area. The DIR and ERA commands allow the use of the full range of file references. For example:

```
DIR *.BAS
```

would list the names of all directory entries on the default drive that have file type BAS.

Transient commands execute in the TPA just as user programs do. A nice feature of CP/M is that, in order to execute any program (system or user), the user merely types its name in response to a CCP prompt. Thus, the runnable version of a program has a file type COM (for command).

There are five important areas addressed by CCP transient commands:

1. Program entry and editing.
2. Utilities such as copying a file from one disk to another.
3. Generating and saving various versions of the operating system.
4. Debugging aids.
5. Language processing.

We will discuss these areas one at a time.

| Command | Action |
|---|---|
| B | Moves pointer to beginning of file. |
| -B | Moves pointer to end of file. |
| ±nC | Moves pointer ±n characters. |
| nFxxx | Places pointer after nth recurrence of string xxx. |
| ±nL | Moves pointer up or down n lines. 0L places the pointer at the beginning of a line. |

*Fig. 3a. Pointer positioning commands.*

| Command | Action |
|---|---|
| ±nD | Delete ±n characters. |
| I | Insert text. |
| ±nK | Kill (delete) ±n lines. |
| ±nT | Type ±n lines on console. 0T types line up to pointer. 1T (or T) types line from pointer to end. |

*Fig. 3b. Basic edits.*

**Entry and Editing (ED)**

A powerful editor (ED) is included in the CP/M operating system. This is a character editor as opposed to a line editor, meaning that a file is considered to consist of one long string of characters with CR (carriage return) and LF (line feed) characters separating each logical line. This string is held in a buffer area in memory. A pointer must be properly positioned in the text to indicate the location of each edit. Some of the basic pointer manipulation commands are shown in Fig. 3a. As an example of the use of these editing commands, the command

```
*B2FXYZ↑Z-3C
```

accomplishes the following:

- Moves pointer to beginning of buffer.
- Positions the pointer immediately after the second occurrence of the string XYZ. Note, ↑Z (control-Z) is used to delimit the string.
- Moves the pointer back three characters, i.e., it is now positioned before the X in the second occurrence (XYZ).

Once the pointer is positioned, a number of edits can be performed. These are listed in Fig. 3b. As an example of the use of these commands, consider the following two edit lines that are equivalent:

```
*B2FXYZ↑Z-3DOTT
*B2FXYZ↑Z-3C3DOTT
```

The first line positions the pointer immediately after the second occurrence of the string XYZ; deletes the three characters preceding the pointer, i.e., the XYZ; and finally prints out the resulting line. The second example first positions the pointer following the second occurrence of XYZ, then moves the pointer back three spaces, finally deletes the XYZ and prints out the resulting line.

Since programs are line oriented, it is useful to be able to perform the basic functions of a line-oriented editor: inserting a line between two existing lines, deleting a line and replacing a line. While these functions are not entirely obvious, they can be accomplished. Assume for

```
A>ED BIG.BAS (CR)
*100A (CR) (bring first 100 lines into buffer)
*(edit first 100 lines)
*100W (CR) (write edited lines to temporary area)
*100A (CR) (bring second 100 lines in)
*(edit second 100 lines)
*E (CR) (end edit)
```

*Example 2.*

```
A>PIP X.ASM = MAIN.ASM,SUB1.ASM,SUB2.ASM,SUB3.ASM
```

*Example 3.*

```
A>PIP TEST.BAS = CON:, X.BAS, Y.BAS
```

*Example 4a.*

```
A>PIP LST: = ONE.ASM, TWO.ASM, THR.ASM.
```

*Example 4b.*

demonstration purposes that the BASIC program shown in Fig. 4 is to be edited. We wish to insert the following line:

```
30 INPUT X
```

and also to replace line 50 by the line

```
50 NEXT I
```

These edit lines accomplish these functions:

```
*BF40↑Z-2CI30 INPUT X
*BFX2YP↑ZOLKI50 NEXT I
```

The first edit line positions the pointer after the 40 and then moves it back two characters so that it is before 40. Then the INPUT statement is inserted. The second example positions the pointer after the erroneous string X2YP, then moves it to the beginning of the line, kills the line and inserts 50 NEXT I. Thus, line replacement is done by first deleting, then inserting the new line. Actually, using this editor does grow on you after some practice, even though it seems complicated at first.

As the icing on the cake, ED has several additional editing commands. For instance, the edit line

```
*BMSFIRST↑ZSECOND↑ZOTT
```

does a search for all occurrences of the string FIRST, replaces each occurrence with the string SECOND and prints out each altered line. The only new editing characters in this line are the S, which is the search command, and the M, which indicates that the next commands are to be repeated as many times as possible, i.e., until the end of the file. If a user is careful, he can perform most desired edits with the Search (S) command.

The above discussion assumes that the file to be edited is located in a memory buffer. The designers of ED were aware, however, that a particular user might not have enough memory to hold an entire program at once. Thus, the editor contains commands that have to do with bringing parts of the file into memory and writing already edited sections to a temporary disk file to make room for another segment of the unedited source. Fig. 5 gives a list of some of these commands. A user with enough memory to hold only 100 lines of BASIC might perform the sequence in Example 2 to edit a 200-line program.

Note that the end edit (E) command does several things. First, it appends any remaining lines in the memory buffer to the temporary file, then it appends any remaining source file lines to the temporary file. Next, it renames the original file, giving it file type BAK (BIG.BAK) for backup purposes. Finally, it creates a file under the original name (BIG.BAS) from the edited temporary file. So part of every editing run is a backup of the original file.

```
10 S = 0
20 FOR I = 1 TO 10
40 S = S + X
50 X2YP
60 PRINT S/10
70 END
```

*Fig. 4. Sample text.*

### Peripheral Interface Program (PIP)

A second major transient command is the PIP program. PIP consists of a number of parts that perform utility functions for the user. One of the basic utility functions available allows the user to make a copy of an existing file. The PIP statement

```
A>PIP NEW.BAS = OLD.BAS
```

will copy the file OLD.BAS on the default disk to a new file called NEW.BAS. A rather interesting extension of this basic idea is to copy several files back to back to a newly created file. The statement in Example 3 could be used to create a program file from a main program (MAIN.ASM) and append three subroutines (SUB1.ASM, SUB2.ASM, SUB3.ASM) called by the main program. This makes a modular approach to programming easy to implement. Just save commonly used subroutines as separate files and, when needed, append them to the main program with PIP.

In order to understand the final application of PIP, we must recall the difference between logical and physical devices. A physical device is just what you would think — a TTY, a CRT, a printer, etc. Logical devices are devices defined in the BIOS, such as CON (console) and LST (list). Logical devices must be assigned to specific physical devices before communication between them is possible. Consequently, the cold-start procedure would be to assign CON to your TTY or CRT and to assign LST to your TTY (normally you would want hard-copy output). This assignment is accomplished by the use of a set of eight front-panel switches called the IOBYTE switches. For example, the switch settings

```
00000001
└┘    └┘
LST  CON
```

accomplish this assignment. The switches not used in this example are for assigning a tape reader and punch; so unless you have such devices, these would always be left in the zero position.

PIP allows the user to refer to these logical devices, and therefore to the corresponding physical devices. If we decided to write a program called TEST.BAS, consisting of a main program to be typed in at the console that calls two subroutines X.BAS and Y.BAS already located on the default disk, we could use Example 4a.

If we simply wanted a listing of ONE.ASM, TWO.ASM and THR.ASM, we could use Example 4b. The colon is necessary to distinguish the logical device name from a disk file.

### System Creation and Maintenance

CP/M contains the software necessary for procreating itself in various forms. A system can be created to accommodate any amount of memory from 16K to 64K bytes in increments of 8K. The transient commands CPM and SYSGEN are necessary to accomplish a change of system size, while SYSGEN alone will make a copy of an existing system. Typically, a user who has just installed a third 8K memory board would use the command CPM 24 * to generate a 24K system. The SYSGEN command is then used to write the newly generated

| Command | Action |
|---|---|
| nA | Append next n lines of source to memory buffer area. n = # implies whole program. |
| E | End edit run. Create edited file. |
| Q | Quit edit run. Make no changes to file. |
| nW | Write n lines from memory buffer to temporary work file. |

*Fig. 5. Text movement editor commands.*

| Option | Meaning |
|---|---|
| A | Enter assembly-language mnemonics. |
| D | Display memory. |
| G | Execute with breakpoints. |
| L | Disassemble |
| M | Move a segment of memory. |
| S | Change memory values. |
| T | Trace program execution. |
| X | Examine CPU state. |

*Fig. 6. DDT command types.*

system onto the first two tracks of the disk in drive B. This system can be given control by placing it in drive A and doing a restart. Thus, it is relatively easy to change system size. Even if the user has only one disk drive, this can be accomplished by modifying the SYSGEN command to write its output to drive A instead of drive B. Exactly how this is done is part of the next subject.

## Dynamic Debugging Tool (DDT)

One of the more surprising transient commands to be found in the CP/M system is DDT. This command has a variety of options that enable the user to interactively execute an assembly-language program. Included in the package is the ability to set breakpoints, single or multiple step through the program, alter the command (runnable) version of a program, disassemble the command version of a program, insert assembly-language statements, examine status flags and more. These capabilities make DDT a useful and powerful part of CP/M. Fig. 6 gives a partial listing of DDT command types.

The customary process for using DDT is first to write and assemble a program so that the command version (file type COM) is available to DDT. The debugging package is then invoked as follows:

```
A> DDT TEST.COM (CR)
```

This command loads DDT into memory instead of CCP, and DDT in turn loads TEST. COM at location $100_{16}$. Now any of the command types can be executed. For example, suppose we desire to test the code shown in Fig. 7a, which writes a 2 out to the front-panel programmed output lights. The assembled version is shown in Fig. 7b. Given that we have invoked DDT, we can illustrate its capabilities with a few examples. First, let's check to see if the program is in memory beginning at location $100_{16}$. This is done in Example 5a and this agrees with the machine-language version in Fig. 7b.

```
JFS   ORG  100H
      MVI  A, 2
      OUT  0FFH
      JMP  0
      END  JFS
```

*Fig. 7a. Sample assembly-language program.*

| Location | Machine Language |
|---|---|
| 0100 | 3E02 |
| 0102 | D3FF |
| 0104 | C30000 |

*Fig. 7b. Machine-language version of TEST program.*

Now let's single step through the program to see if it performs its intended function (See Example 5b). The Trace command gives the state of the CPU, as indicated by the carry (C), zero (Z), minus (M), even parity (E) and auxiliary carry flags (I), the contents of the registers (A, B-C, D-E, H-L); the contents of the stack pointer (S), the program counter (P); the mnemonics of the instruction at the location pointed to by P (i.e., the instruction to be executed next); and, finally, the location from which the following instruction will be taken ($0102_{16}$). Let's take another step in Example 5c.

Here, the MVI A,2 instruction has been executed so the A register is changed accordingly. None of the status flags have changed. The instruction about to be executed is the OUT, OFFH, and the next instruction will come from $0104_{16}$. To finish the program, one more step (Example 5d) is required. Here, the program returns control to the operating system via JMP 0. The program seems to work properly.

Two examples of the other command types are as follows:

```
-L100, 106 (CR)
0100 MVI A,02
0102 OUT FF
0104 JMP 0000
```

The disassemble command recreates the assembly-language mnemonics.

```
-A100 (CR)
0100 MVI A,01 (CR)
0102 (CR)
```

This sequence replaces the MVI instruction by MVI A,1. Assembly-language statements can thus be entered at any location in the program. DDT takes care of assembling such statements. Finally, the

```
-D100,10 6
0100 3E 02 D3 FF C3 00 00
```

*Example 5a.*

```
-T (CR) (trace one step)
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0100 MVI A,02 *0102
```

*Example 5b.*

```
-T (CR)
COZOMOEOIO A=02 B=0000 D=0000 H=0000 S=0100 P=0102 OUT FF*0104
```

*Example 5c.*

```
-T (CR)
COZOMOEO1O A=00 B=0000 D=0000 H=0000 S=0100 P=0108 JMP 0000*0000
```

*Example 5d.*

following sequence changes back to the original MVI instruction by changing the $01_{16}$ in location $101_{16}$ to an $02_{16}$.

```
-S100
0101 01 02 (CR)
0102 D3 .(CR)
```

Note that the period ends the substitute mode.

As is readily apparent, DDT offers an invaluable tool for debugging assembly-language programs.

**The Language Processors**

The two main languages supported by CP/M are 8080 assembly language and BASIC. The assembler (ASM) interacts with CP/M as follows. Utilizing the editor, the user creates an assembly source file, say TEST.ASM, on disk. This file is assembled via the transient command ASM TEST. There are two important outputs of the assembly process. The results of the assembly, including error messages, are placed into a file named TEST.PRN. These results can be viewed via the TYPE TEST.PRN command. The other output is a disk file named TEST.HEX, which contains the machine-language output of the assembly. The LOAD TEST command now is invoked to create a new file named TEST.COM, which contains the binary (runnable) version of the program. This version of the program can be tested simply by typing its name as a CCP command or via DDT as described above.

```
ED $1.ASM
ERA $1.BAK
ASM $1
TYPE $1.PRN
ERA $1.PRN
LOAD $1
$1
```

*Fig. 8. SUBMIT file for editing, assembly and tests.*

This whole process of editing, assembling, loading and running is such a common sequence that it would be helpful to be able to teach CP/M to do the whole sequence by itself. In fact, this can be accomplished using the concept of SUBMIT files. A SUBMIT file is a disk file of CCP commands, except that the specific names (or name) of the parameters are left unspecified. Instead, they are represented by $1, $2, etc. Fig. 8 shows a listing of a SUBMIT file named AS.SUB that is useful for the above editing, assembly and test process. To instruct CP/M to execute this SUBMIT file, the user simply types the transient command

```
A>SUBMIT AS TEST
```

All occurrences of $1 are replaced by the first parameter in the parameter list (TEST), and CPM executes the list of commands as though they had been typed individually. SUBMIT files give CP/M a capability similar in nature to the job-stream concept in larger machines.

The CP/M BASIC is a full version of BASIC with floating-point arithmetic and the full complement of built-in numeric and character-handling functions. It takes 20K to run the BASIC language processors.

The procedure for running a BASIC program is to first create a disk file of BASIC source statements, say TEST.BAS. The BASIC-E TEST transient command does a partial compilation of the source file, producing an intermediate file called TEST.INT. The RUN-E TEST command is used to load and run the program.

**Conclusion**

The intention of this article has been to present enough details of the CP/M operating system to give the reader a flavor for the degree of sophistication of currently available software.

It is an interesting intellectual exercise to think about writing one's own operating system, but it seems clear that with such sophisticated software available at a reasonable price, the time and cost of writing an operating system is prohibitive.

No comparison has been attempted between CP/M and similar software products on the market. It's difficult enough to keep track of the names of all the companies dealing in various aspects of the microcomputer market. It would take a great deal of time to evaluate all the software competitive with CP/M. I hope this article will offer a friendly challenge to others knowledgeable in particular micro operating systems. Let's see an article or two on these other systems. Let's bring micro-systems software out in the open. The personal effort is worthwhile and would be instructive to us all. ■


**References:**

*Imsai CP/M Floppy Disk Operation System Version 1.31, Rev. 0*, 1976, Imsai Manufacturing Corporation, San Leandro CA 94577.

*An Introduction To CP/M Features and Facilities*, 1976 (this and all following refs. by Digital Research, Pacific Grove CA 93950.

*ED - a Context Editor for the CP/M Disk System, User's Manual*, 1976.

*CP/M Assembler (ASM) User's Guide*, 1976.

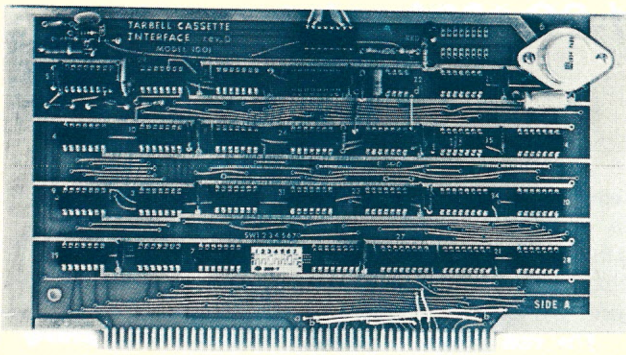
*CP/M Interface Guide*, 1976.

*CP/M System Alteration Guide*, 1976.

# TARBELL SETS STANDARDS

## For Hobbyists and Systems Developers

Sales to thousands of hobbyists over the past two years have proven the Tarbell Cassette Interface to be a microcomputer industry standard. Tarbell Electronics continues research and development to produce new and efficient components to fill hobbyists' changing needs.

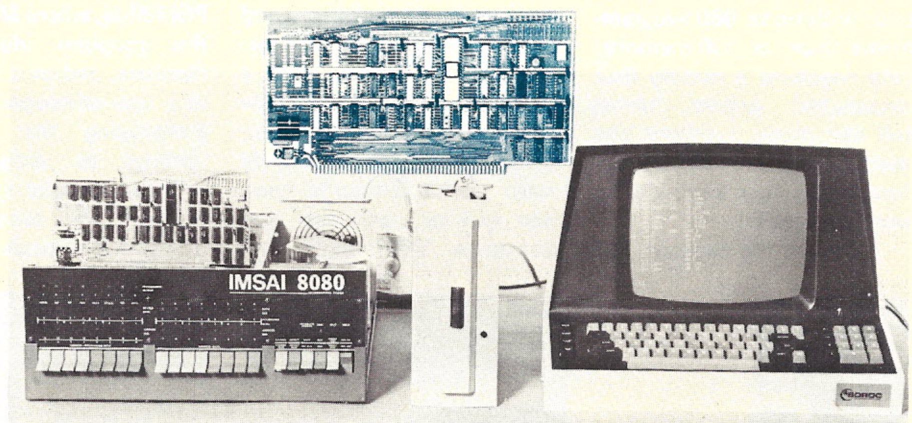



## TARBELL CASSETTE INTERFACE

- Plugs directly into your IMSAI or ALTAIR*
- Fastest transfer rate: 187 (standard) to 540 bytes/second
- Extremely Reliable—Phase encoded (self-clocking)
- 4 Extra Status Lines, 4 Extra Control Lines
- 37-page manual included
- Device Code Selectable by DIP-switch
- Capable of Generating Kansas City tapes also
- No modification required on audio cassette recorder
- Complete kit $120, Assembled $175, Manual $4

## TARBELL FLOPPY DISC INTERFACE

- Plugs directly into your IMSAI or ALTAIR* and handles up to 4 standard single drives in daisy-chain.
- Operates at standard 250K bits per second on normal disc format capacity of 243K bytes.
- Works with modified CP/M Operating System and BASIC-E Compiler.
- Hardware includes 4 extra IC slots, built-in phantom bootstrap and on-board crystal clock. Uses WD 1771 LSI Chip.
- 6-month warranty and extensive documentation.
- PRICE:
  Kit $190 . . . . . . Assembled $265

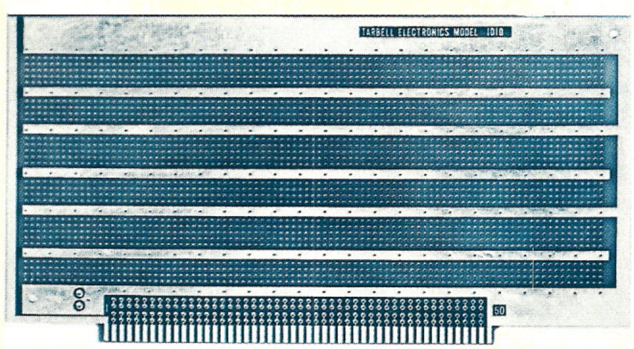



**Compatible Disc Drives**

Ask about our disc drives priced as low as $525.

## TARBELL PROTOTYPE BOARD

**Model 1010**




- Gold plated edge pins
- Takes 33 14-pin ICs or
- Mix 40-pin, 18-pin, 16-pin and 14-pin ICs
- Location for 5 volt regulator
- Suitable for solder and wire wrap
- ALTAIR/IMSAI compatible

**Price:** $28.00

For fast, off the shelf delivery, all Tarbell Electronics products may be purchased from computer store dealers across the country. Or write Tarbell Electronics direct for complete information.

*ALTAIR is a trademark/tradename of MITS, Inc.

Tarbell Electronics

20620 South Leapwood Avenue, Suite P
Carson, California 90746
(213) 538-4251

T11

*Arthur G. Burns*
*7100 NW 17 St.*
*Bldg 3, Apt 315*
*Plantation FL 33313*

# Space-Saver System

## *the TI 59 programmable calculator and PC-100A printer*

Calculator — my cost $395, your cost $35. With that "for sale" ad successfully (?) transacted, I announced to my still-patient spouse (a veteran of H-P 35s, TI 51 and TI 52 calculator campaigns), "Now I'm going to get *the* calculator!" *The* calculator was to be a TI 59 programmable plus a PC-100A printer. "Think of it," I exclaimed. The $395 price tag that five years ago had hung on my first hand-held "number-cruncher" would now include: a magnetic card or keyboard-programmable portable calculator with up to 960 programmable steps, or 100 memory/data registers; a mating (but detachable) printer having full alphabetic, symbolic and numeric typeset. The printer functions under keyboard or program results!

I bought the package, and I still think it was a super purchase. Here's an armchair tour of what I found.

### Solid State "Software"

A new innovation emerging in hand-helds is the use of solid-state chips containing multiprogram libraries the user can insert and remove from the calculator. The TI 59 comes with a Master Library Module "chip" containing 25 programs ranging from matrix math (a 9 x 9 matrix inversion can be performed that occupies 898 steps and requires 12 minutes to solve!) to moving averages, compound interest, annuities, etc., and yes, of course, a game (HI-LO)!

The real power of these library modules is their easy accessibility through a simple keyboard call-up routine, 2nd PGM-M-N, where M and N are the program identification numbers, and/or a subroutine in a user-developed program. Employing the solid-state libraries as program subroutines actually extends the program step capability out into the thousands of steps in many cases.

### Manuals — Back to the Books

The TI 59 comes with two large (8½ x 11) manuals, *Personal Programming* and *Master Library*. If you're the "push the switches and buttons and read later" type, these widgets will be your Waterloo. *Personal Programming* was my evening reading material for two solid weeks! There are 45 keys on the TI 59 keyboard. Through their direct function, and when combined with the 2nd and INV keys, they allow 108 operations from the keyboard! The manual's large print and organization of instructions are effective if a reader sequentially works his way through it. However, a ring-binder type of manual instead of the hard binding type used would help the reader.

*Master Library* details the key usage sequence and gives sample problems for each of the 25 programs stored in the Master Library solid-state module. A disappointment was the lack of a full step sequence listing for any of the programs. Such a listing would help the novice programmer understand how program sequences are optimally employed. For owners of the TI 59 and PC-100A printer, an answer does exist: a down-loading procedure that transfers a selected library routine into the calculator memory where it can then be printed out as a step-by-step sequence.


*No need to build an extra room in the house for this combo!*

*Programming a future article?*

**Programming, Calculator Style**

Flowchart problem-structuring, subroutines, GOTO, branching, looping, conditional testing and transfer, terms familiar to mini and microcomputer users, are applicable to TI 59 programming. With a couple of keystrokes, the TI 59 can be shifted back and forth from the calculator mode to the programming mode, wherein each keystroke can be stored in the calculator's memory and, if desired later, onto magnetic cards for permanent storage.

Another innovation is the TI 59's ability to allocate or partition the total calculator memory between program steps and data storage registers. Starting with 100 memories and 160 program steps, you can trade in blocks of ten program memories to gain 80 program steps. The maximum is 960 steps.

The partitioning is performed easily from the keyboard and adds to the calculator's versatility. For example, some games, such as Blackjack, require many program steps and few memories. Other uses, such as stock-market 30-or-60-day moving averages require lots of data memory and not many program steps.

**Put It in Writing — Print It!**

Hours and hours of programming and debugging on a TI 52 had convinced me that my next calculator would have to have printout capability. The TI 59 exceeded my expectations for ease of programming and clear presentation of program results. The PC-100A printer incorporates a complete alphabetic, symbolic and numeric typeset. Full program titles and prompting directions can be printed, and calculation results can be labeled.

Don't expect to use this feature too liberally, however, since it's quite costly in terms of programming space. Each letter, number, space or symbol used has to be coded in as a two-digit number. The printer uses 2½-inch-wide heat-sensitive paper and solid-state heating element typeset. Twenty characters (alpha or numeric) can be placed on one line. Operation is *really* whisper-quiet.

**Primary Modes of Operation**

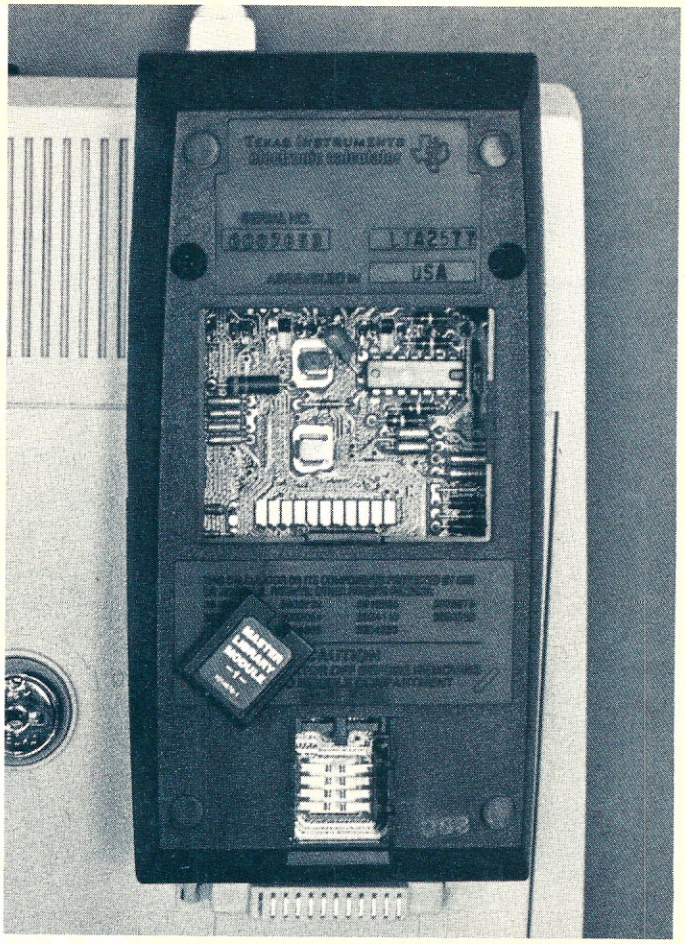
- List — Prints out each program step number and a keycode number that identifies which key was pressed for each step of the entire program.
- Trace — Prints out every calculation value and the instruction that generated that value.
- Calculation printout — This is under program control and causes a printout of intermediate and final calculation values. Four character labels can be added to each line.
- Plot — Also a program-controlled feature allowing the printer to print an asterisk at any of 20 locations across the tape width. Since the tape advance can also be under program control, the result is a handy but rudimentary plotting capability. The personal programming manual shows a sample sinewave plotting program.

**More to Come**

In addition to the Master Library Module, which comes as standard equipment with the calculator, TI has other solid-state modules on real estate, statistics, navigation and surveying. These can be purchased separately and readily substituted for the Master Library.

A user's group, sponsored by Texas Instruments has formed for TI 59 owners; its purpose is to encourage program exchanges. The calculator is so new, and the user's group response has been so huge, that it apparently caught TI by surprise — so details of program listings, etc., are still at the printers. I'm sure *Kilobaud* readers will be interested in programs of applications, games and unusual printouts since this is unquestionably going to be a "hot" user combo.

Write, publish, and save your money, because although this certainly is *the* calculator, PET computer literature sure looks terrific. ■



*View of the TI 59 from the back. Removing the two-cell battery accesses the printer interface contacts. Note the Master Library Module, which contains 25 programs.*

# Everyone's getting personal in Long Beach.

3 full days of technical sessions, exhibits, home-brew displays and the latest on personal and small business computing, all at PERCOMP '78. April 28-29-30.

Just for the fun of it, we have an entire home-brew section... robotics, games, computer music, even every-day, sensible stuff like checkbook balancing and preparing mailing lists. You're sure to take home some new tricks to your computer.

And don't forget, PERCOMP '78 has booth after booth of everything in personal and small business computing.

5 months before show time our dynamite exhibit list includes from A to V:

**The Astute:**
Advanced Computer Products
Alpha Supply Co.
Apple Computer, Inc.
A-Vidd Electronics

**The Brilliant:**
Byte Industries Incorporated
Byte Shop Lawndale
Byte Publications, Inc.

Jim Butterfield is on his way from Toronto with the entire, unabridged truth about KIM. Jim co-authored *The First Book of KIM.*

Attorney Kenneth Widelitz will be on hand with some friendly advice on "Tax Aspects of Lemonaide Stand Computing" while his friend attorney Leonard Tachner delivers the low-down on "Patents, Copyrights and Computers."

Carol Anne Ogdin's keynote address bares the facts on "How Personal Computers Are Being Used Today." Carol comes to us from Software Technique, Inc. in Alexandria, Virginia.

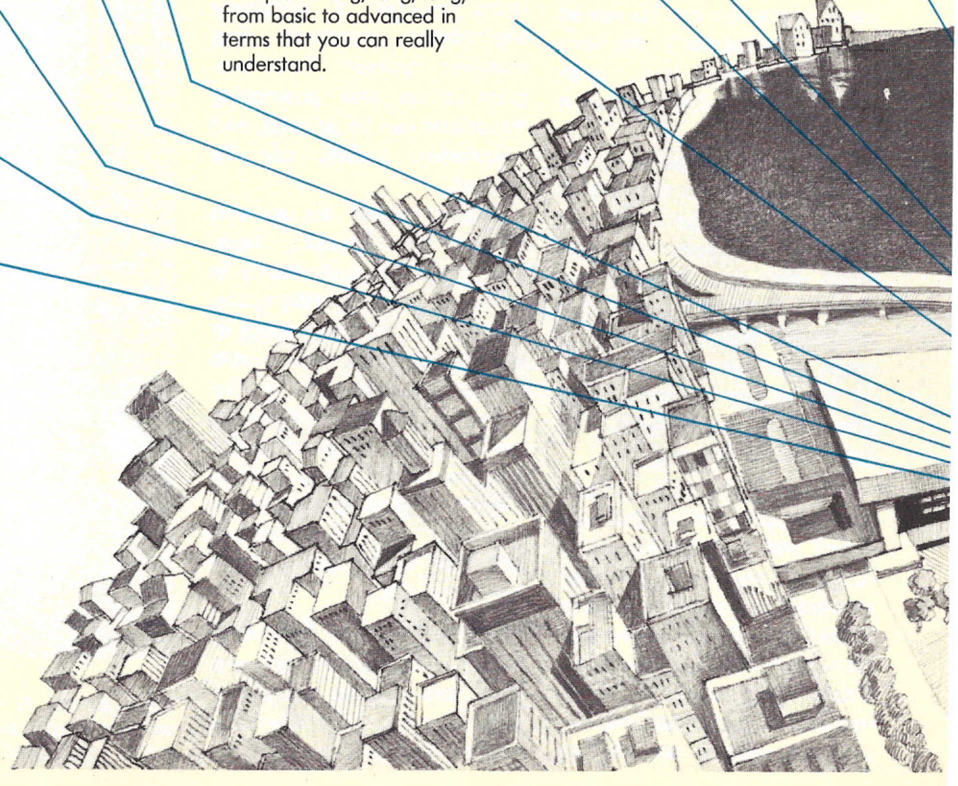
Admission for 3 full days of personal computing, complete with 180 exhibits, 66 fascinating seminars and all the going and coming you want is $10 ($8 for students and juniors) at the door, and $8 ($6 for students and juniors) if you pre-register.

Dr. Portia Isaacson, a contributing editor for Datamation and an associate of Byte, brings computer enthusiasts the very latest word on "Computer Store Retailing."

Whether you're just a beginner or a well informed expert, you'll find the latest on ham radio communications, graphic systems, word processing, pattern recognition or...(our list of topics is long, long, long) from basic to advanced in terms that you can really understand.

Louis Field, president of the International Computer Society/SCCS, gives you everything he's got on "Getting Started in Micro-Computing."

From Creative Computing Magazine comes David Ahl with all you'll ever need to know on "Marketing for the New Manufacturer."



**The Captivating:**
Calcomp
Computalker Consultants
Computerland
Computer Magazine
The Computer Mart of Orange
Computer Pathways Unlimited, Inc.
Computer Power & Light Inc.
Creative Computing

**The Dynamic:**
Databyte, Inc.
D.C. Hayes Assoc.
Dilithium Press
Dynabyte, Inc.

**The Energetic:**
Edwards Assoc.
Electronics Warehouse, Inc.
Electro-Sonic Components, Inc.
Entech

**The Hearty:**
Heathkit Electronic Centers
Hobby World

**The Irresistible:**
Interface Age Magazine

**The Jovial:**
Jade Company
James Henry Co.

**The Keen**
Kathryn Atwood Enterprises
Kilobaud Magazine

**The Magnificent:**
Marinchip Systems
Micropolis Corporation
Mission Control

**The Omnipotent:**
OK Machine & Tool Corp.
Olson Electronics, Inc.
Optical Electronics, Inc.
Orange County Computer Center

**RIP THIS COUPON FROM THE PAGE AND GET IT TO US BY APRIL 10.**

I want to save time and money.
Please send me_______ pre-registration forms.

**PERCOMP '78**
1833 E. Seventeenth St., Suite 108, Santa Ana, Ca. 92701
(714) 973-0880 K4

Name____________________________________

Address____________________________________

City__________________ State________ Zip________

**The Personable:**
Pan Dynamics, Inc.
Personal Computing
Problem Solver Systems, Inc.

**The Quintessential:**
Quainco Ltd.
Quest Electronics

**The Remarkable:**
Radio Shack
ROM Publications, Corp.

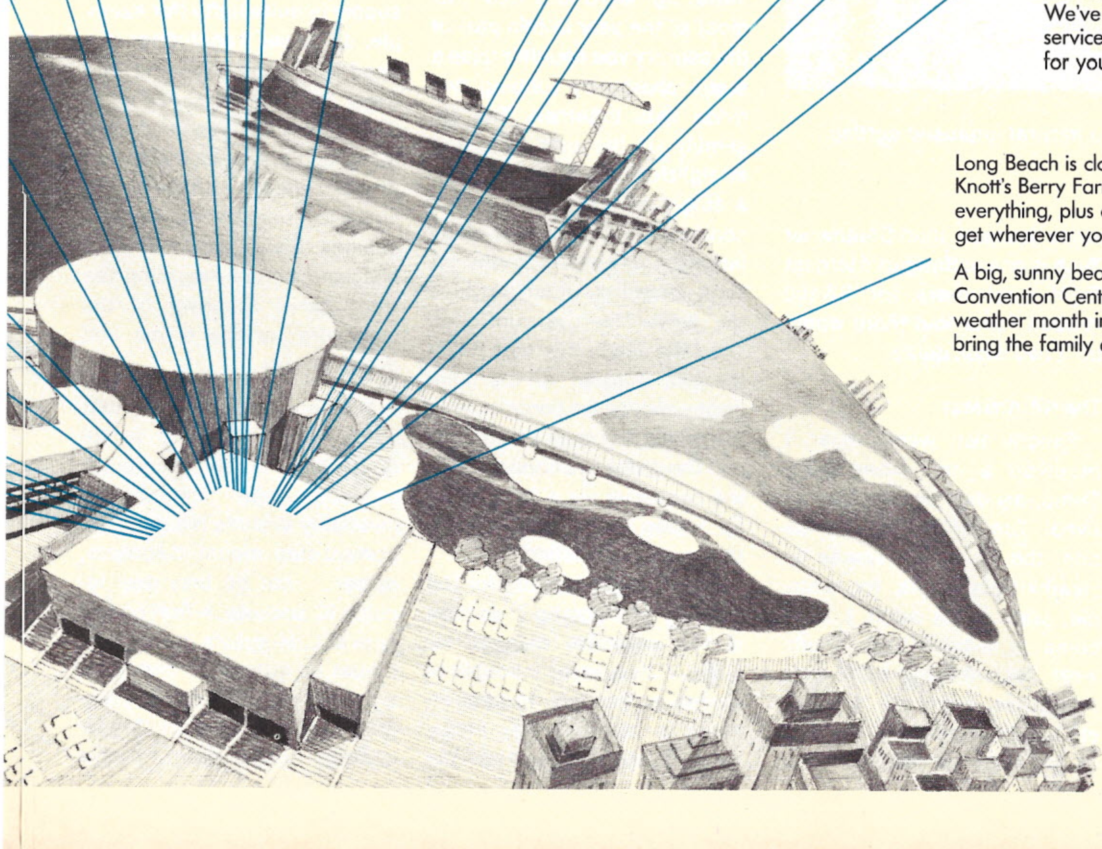
**The Sterling:**
SD Sales
Space Byte Corp.
SubLogic Co.
Sunshine Computer Co.
Sybex, Inc.
Szerlip Enterprises

**The Tantalizing:**
Tandy Computers
Tarbell Electronics
Tech-Mart
Telpar, Inc.
TLF, Corp.

**The Ultra:**
Ultra-Violet Products, Inc.

**The Valiant:**
Vector Graphics, Inc.
Vista Computer Co.

Since everybody's coming, better make your advanced reservations. Pre-register and save (you won't have to wait in line) ...but don't forget about your hotel room. Our staff has reserved rooms in hotels and motels near the Convention Center. We've even arranged for a shuttle bus service. So call and we'll save a room for you.



Long Beach is close to Disneyland, Knott's Berry Farm, Universal Studios... everything, plus our staff will help you get wherever you want to go.

A big, sunny beach is minutes from the Convention Center, and April is a great weather month in Long Beach, so plan to bring the family and have a good time.

*Mickey Ferguson*
*PO Box 708*
*Trenton GA 30752*

# How to Make Your SWTP System Happy

## *give it a couple of floppies!*


*The author and his system in a natural, unposed setting.*

Last September, our local Southwest Tech dealer received his first MF-68 Minifloppy Disk System. I had been considering the purchase of a floppy-disk system for several months and was sufficiently impressed with the MF-68 that I immediately gave him a check. The MF-68 has dual drives, controller/interface, FDOS software, Disk BASIC and a disk-based co-resident editor/assembler, all for only $995. Before I left Data-Comp (our local dealer), they called Southwest Tech to place the order and were told the demand was unexpectedly large. (We have since learned that Southwest Tech is one of Shugart's largest OEM customers for SA400 drives.) I was told there would be a two-week delay.

### The Kit Arrives

Exactly two weeks later, I received a call from Data-Comp; my disk system had arrived! The MF-68 documentation, though rather sparse by Heathkit standards, for example, was about what I have come to expect from Southwest Tech (having built their computer, terminal, printer, etc.). A booklet about the size of an issue of *Kilobaud* discussed the software, and another smaller booklet contained the assembly instructions. Setting aside the software book, I began reading the assembly manual, which contained typical Southwest Tech instructions—mount resistors, solder, mount capacitors, solder, etc. The manual also contained dire warnings, which I promptly ignored, about handling MOS devices (for most of the year in this part of the country you couldn't raise a static charge by rubbing two nylon cats together). The assembly of the unit appeared straightforward. The controller, a single printed circuit board, contains thirteen ICs and plugs into I/O port 6 in the Southwest Tech computer, which provides the power for the controller. The little black box that holds the two Shugart SA400 minifloppy disk drives contains only the drives and their power supply. The drives are connected to the controller via a 34-conductor ribbon cable with connectors already attached.

I then began looking for the "Theory of Operation" section in the assembly manual. Instead, I discovered a couple of sentences that refer to some articles in *Interface Age* magazine—if you must know how the Western Digital 1771 floppy-disk controller chip on the controller card works. And I don't even know anyone who gets *Interface Age*! Then I turned to the "In Case of Difficulty" section, where I was informed that if it doesn't work, box it well and ship it back. Sure hope this thing works!

### The Assembly

The actual assembly of the MF-68 was easier than the instructions had indicated. The unit went together with less hassle than any Southwest Tech kit I have built. This really says a lot because, as I said, I have built several of their kits. The only problem encountered during assembly involved a .1 uF capacitor destroyed in shipping. Also, someone forgot to include the wire ties to make the power supply for the disk drives a little neater. I replaced the capacitor with one from my junk box and purchased the needed wire ties from Radio Shack.

The controller (see Photo 1) is a rather small, quickly assembled printed circuit board. However, because of the amount of hand-wiring required, the assembly of the power supply takes considerably longer. The power supply is quite hefty (for example, its power transformer is a good bit larger than the one used by Southwest Tech in their computer). Working slowly and taking a break for lunch, I needed approximately six hours to assemble the MF-68. I decided to wait until the initial checkout before mounting the front panel or cover of the enclosure for the drives and power supply.

My only complaint, to that point, applies equally to all Southwest Tech computer products—their instructions invariably exclude the most important step: mount IC sockets, solder . . . nor do they ever include IC sockets. A firm believer in M L (Murphy's Law), I *only* mount an integrated circuit on a printed circuit board by plugging it into a socket! Consequently, I never seem to have any problems with defective integrated circuits.

### The Moment of Truth

With my MF-68 assembled and all connections double-checked, I was ready to apply power. A check of the power supply revealed the presence of 12 volts, 5 volts and ground required on the proper pins of the plugs that connect to the drives. I loaded the bootstrap program provided by Southwest Tech into the computer (and saved a copy to tape) and ran the bootstrap.

Nothing happened! Back to the instruction book, where I find that it may be necessary to run the bootstrap a couple of times before it brings the system up. After a second try, nothing happened. Four hours later, I found no unusual voltages and I knew the clock for the WD1771 controller chip was working; but the bootstrap wouldn't bring the disk system on line! So I loaded the MF-68 into the old VW and headed for Data-Comp, where we visually checked the system and found no errors.

We tried to bootstrap their system with my controller and their drives. The system came on-line on the first attempt. We then tried the system with my controller and drives; again it worked the first time! Obviously I had left the problem at home in my computer! Since we had my MF-68 system running on Data-Comp's computer, we decided to give it a thorough check. We found that only one drive was working (bet you thought this sort of thing never happens when someone builds a kit with the idea of writing a magazine article about it). Drive 0 was functioning perfectly, but the read/write head of drive 1 was stuck in the Home position and refused to move. There was no way for drive 0 to avoid working; it was serial number 6800! Data-Comp did not have a replacement drive; therefore, I shipped the bad one back to Southwest Tech for a replacement, which I received nine days later via parcel post.

At home, I removed the motherboard from my computer and began examining it. The malfunction stuck out like a sore thumb—a single unsoldered connection on the underside of the motherboard. This unsoldered pin was on the molex connector for I/O port 6 (the one used for the controller board). With the connection soldered and the system reassembled, the unit bootstrapped on the first try—everything except the defective drive worked perfectly.

Why have I told you the details of my problem in getting my MF-68 up and running? Because I wanted to illustrate the old trap of instinctively blaming the new component without checking the rest of the system for a possible cause. After discussions with many MF-68 users in different parts of the country, I found the most common problems involved the motherboard, where cold solder joints or unsoldered connections usually appear. Some people have had problems with the PACK and/or COPY functions of the DOS—usually traced back to a marginal IC on the motherboard. The other problems include defective crystal on the controller board, which affects the clock for the WD1771, and defective disk drives. However, the systems of the vast majority of MF-68 owners with whom I have spoken, have worked properly the first time power was applied.

### The Software

Southwest Tech FDOS V1.0, written by Robert Uiterwyk, recognizes the following commands: CATALOG (which may be shortened to CAT), HOME, FILES, PRINT, SAVE, LOAD, RUN, CREATE, INIT, DELETE, PURGE, RENAME, EXIT and TEST. All FDOS commands use the same format—command drive number, file name and password (if used). If the drive number is not specified, it is assumed to be zero. FDOS commands follow in detail.

**CATALOG**. Lists the names of all files stored on a disk.

**FILES**. Lists the names of all files stored on a disk along with all pertinent information about the file. This information includes: the track and sector address of the file on the disk; the number of sectors used to store the file on the disk; the file type (which may be blank file, system file, object program file, BASIC program file, co-resident editor/assembler source file or BASIC data file); the lowest and highest memory address used by the file; and the entry point (initial program counter value) of the file.

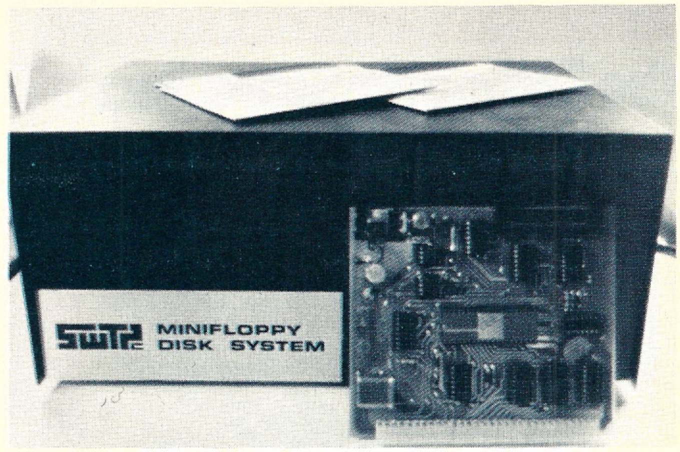



*Photo 1. The entire Southwest Tech MF-68 floppy-disk system. The printed circuit board is the controller card which uses the Western Digital 1771 floppy-disk controller chip.*

**PRINT**. Causes the output generated by the following command to be printed on a Southwest Tech PR-40 printer (at port 7) rather than on the control terminal. The command following the PRINT command should be either CATALOG or FILES.

**SAVE**. Used to store a memory-resident object program on a disk. The SAVE command allocates 25 percent more disk space than is actually required to store the program to allow for future expansion of the program under the same file name.

**LOAD**. Reads an object program from a disk file into the computer's memory. After the program is loaded, system control is returned to FDOS. To run the program, get into MIKBUG and type 'G'. This command is helpful when you have a subprogram used by several other programs but not executed by itself; for example, a random number generator that may be used by many different games.

**RUN**. Loads an object program as in the LOAD command, then automatically executes the program.

**CREATE**. Names a file and allocates disk space for it. CREATE is very useful in making efficient use of disk space. If you were to fill a disk by SAVEing programs, you would waste 25 percent of the disk space. If you are SAVEing programs that will not be expanded, it is advantageous to CREATE the file before using the SAVE command, which uses only the disk space necessary to store the program.

**INIT**. Since the Southwest Tech MF-68 is soft-sectored in the IBM format, considerable information must be stored on a disk before it can be used. This is taken care of when you INITialize the disk. Also, the INIT command stores a copy of the FDOS on the disk.

**DELETE (or PURGE)**. Removes a file from the disk catalog only. To erase the file from the disk, PACK the disk after using the DELETE or PURGE commands.

**RENAME**. Allows file names to be changed. RENAME may also be used to change a password-protected file to an unprotected file or to give password protection to an unprotected file.

**HOME**. Moves the read/write head to track 0.

**EXIT**. Gives system control to the MIKBUG ROM. This command caused me some prob-

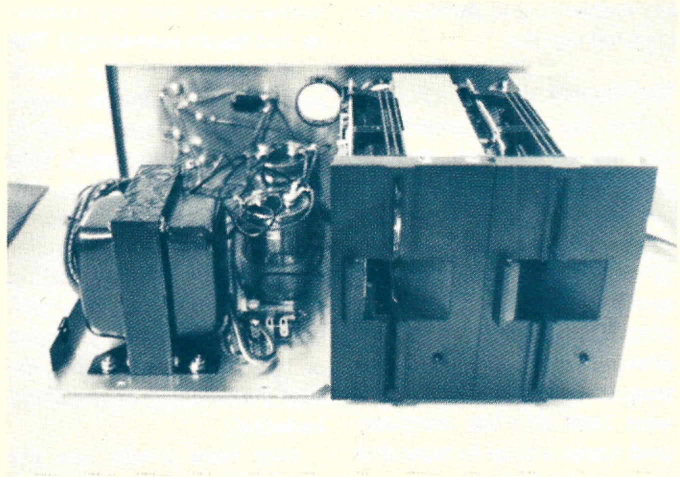


*Photo 2. The MF-68 with its cover and front panel removed. Notice the husky power transformer & filter capacitors.*

lems initially because I use RT-68/MX ROM instead of MIKBUG, and EXIT jumps to the cold-start entry of MIKBUG. The cold-start entry point of RT-68/MX is at a different address.

**TEST**. Reads all tracks and sectors on a disk and verifies the cyclic redundancy check (CRC) numbers. The CRC numbers correspond to checksums on tape. By using the TEST command after the SAVE command, you can find errors before they become problems and repeat the SAVE (which should not fail a second time, unless the disk itself is bad). Although I have never had a bad SAVE, I use the TEST command following the SAVE command.

### System Files

Five system files are provided with the MF-68 FDOS: DOS, PACK, COPY, BASIC and CORES. Compared to the other files, system files are special since they cannot be deleted and are loaded and executed by typing their file name. In a file listing provided by the CATALOG command, system files are identified by a preceding $ (for example: $BASIC). Although the MF-68 documentation provided by Southwest Tech does not discuss how to SAVE or DELETE system files, we will discuss the procedure. First, let's look at the system files provided with the MF-68.

**DOS**. The Southwest Tech Floppy Disk Operating System.

**PACK**. Moves all files on a disk into contiguous sectors, effectively erasing all files DELETEd or PURGEd from the disk. PACK can only be used on drive 0.

**COPY**. Duplicates the contents of the disk in drive 0 on the disk in drive 1. The new disk (the one in drive 1) must have been INITialized prior to the COPY.

**BASIC**. An interim version that should have been replaced with the final version by the time you read this. SWTP Disk BASIC Version 1.0 is essentially SWTP 8K BASIC Version 2.0 with extensions for use in a disk environment. *It does not have data files.* However, the final version will have sequential and random-access data files, plus a few other goodies. Let's look at the extensions to FDOS BASIC Version 1.0 over 8K BASIC Version 2.0 (SAVE, LOAD, TSAVE, TLOAD, DOS, CATALOG, CHAIN, STRING).

*SAVE*. Stores a BASIC program from memory to a disk file.

*LOAD*. Reads a BASIC program from a disk file to memory.

*TSAVE*. Copies a BASIC program from memory to tape and is identical to the SAVE function of 8K BASIC.

*TLOAD*. Reads a BASIC program from tape to memory and is identical to the LOAD function of 8K BASIC.

*DOS*. Loads the SWTP FDOS operating system from disk and gives system control to it.

*CATALOG*. Lists the names of all BASIC (and blank) files that are on a disk.

*CHAIN*. Loads a BASIC program from a disk file and executes the BASIC program. CHAIN may be used as a command (without a line number) for immediate execution or as a statement (with a line number) to allow one BASIC program to call another.

*STRING*. Used to determine the maximum length of string variables. According to Southwest Tech's manual, the maximum string length may be any integer value between 6 and 128. However, we have found that 72 is the maximum value BASIC will accept. With any value greater than 72, BASIC will set the string length to 72. If the STRING = command is not used, the maximum string length is assumed to be 32.

**CORES**. The Southwest Tech co-resident editor/assembler with extensions for use in a disk environment. The SIZE command of the tape version is no longer supported, but the following commands have been added.

*SAVE*. Copies a source program from memory to a disk file.

*LOAD*. Reads a source program from a disk file to memory.

*TLOAD*. Reads a source program from tape to memory and is identical to the LOAD command of the tape version of the co-resident editor/assembler. There is no TSAVE command in the disk version of CORES.

*DOS*. Reads the SWTP FDOS program for a disk to memory and gives system control to it. Under CORES, when assembling a program and using the necessary options, the object program is stored on a disk file rather than on tape. As a result, the second pass of the assembler is considerably faster than in the tape version (particularly if you are not also generating a source listing)!

### Adding, Deleting and Modifying System Files

As I mentioned earlier, instead of the MIKBUG ROM, I use the RT-68/MX by Microware, a great little monitor/real-time operating system (which I've written up in another article). However, it is not 100 percent MIKBUG compatible, which caused me one minor problem with the FDOS. Also, my terminal decodes a non-standard control character as the cursor back space. I prefer to have the cursor actually back up rather than have an underscore echoed when the software receives a back-space command like Southwest Tech's FDOS, CORES and BASIC.

In light of these two factors, coupled with my wish to place several programs on disk as system files, I wanted to be able to add, delete and modify system files. Since Southwest Tech does not provide this information, I will give it to you here. I am indebted to James Caldwell, K5OHU, of the International 6800 User Group for the following information.

To delete a system file use the RENAME command. To delete the system file BASIC, for example, the sequence in Program A would be used (operator inputs are underscored to distinguish from machine prompts).

The procedure for saving a system file differs slightly from that for saving a regular object file. Let's assume we have BASIC in memory, have modified it and now want to save the modified version to disk as a system file. We use the following procedure:

```
FDOS READY
SAVE $BASIC(cr)
FIRST ADDRESS? 0100(cr)
LAST ADDRESS? 23FF(cr)
PROGRAM START? 0100(CR)
? $$(cr)
FDOS READY
```

In the above procedure, the modified BASIC will replace the old system file BASIC at the same place on the disk where the old system file BASIC was located. It is not necessary to first delete the old system file prior to saving the new one. Since the same file name is used for both, the DOS will store the new one at the same location (this is true of any file, not just system files). You can, of course, use the above procedure to add any new system files that you desire.

### The Western Digital 1771

As I have mentioned, the Southwest Tech MF-68 uses the Western Digital 1771 floppy-disk controller chip. The selection of the 1771 was a wise decision for a number of reasons: The 1771 appears to be a microprocessor whose sole function is to allow computers to communicate with floppy-disk systems. The use of the 1771 makes it easier for the person who writes disk-operating system software and enhances the data transfer rate between computer and disk or between disk and computer. The data transfer rate is increased because the computer is, for the most part, freed from worrying what the disk is doing. Suppose the DOS software has determined (by reading the disk catalog) the program it wants stored on the disk at track 5 and sector 6, and the program requires seven sectors of disk space. The computer would then tell the 1771 to go to track 5, sector 6, and get seven sectors; then the computer only has to look for the data to start coming and begin storing it in memory. In some disk systems the computer has to position the head, find the proper sector, etc. This slows the data transfer rate considerably.

```
FDOS READY
RENAME $BASIC(cr)
NEW FILE NAME? MUD(cr)
FDOS READY
DELETE MUD(cr)
FDOS READY
```

(cr) signifies the termination of input with a carriage return

*Program A.*

The 1771 is an MOS/LSI device performing the functions of a floppy-disk controller/formatter, designed to be included in the disk drive electronics, and contains a flexible interface organization that accommodates the interface signals from most drive manufacturers. It is compatible with the IBM 3740 format. The processor interface consists of an eight-bit bidirectional bus for data, status and control word transfers. The 1771 operates on a multiplexed bus with other bus-oriented devices.

Some of the features of the Western Digital 1771 are: soft-sector format compatibility, automatic track seek (with verification), single/multiple record read with automatic sector search, entire track read, fixed or variable record length, single/multiple record write with automatic sector search and entire track write for disk initialization. All of the communications with the data bus for transfers of data or control/status information are double-buffered within the 1771, which is used for programmed data transfers, or in a DMA environment.

The 1771 also contains CRC logic used to generate or check the 16-bit cyclic redundancy check numbers. It *also* contains an arithmetic logic unit (ALU), an address mark detector and timing and control logic.

The language of the 1771 consists of eleven commands, which are RESTORE, SEEK, STEP, STEP IN, STEP OUT, READ COMMAND, WRITE COMMAND, READ ADDRESS, READ TRACK, WRITE TRACK and FORCE INTERRUPT. Without discussing these commands in detail, I will point out that they are essentially concerned with positioning the read/write head or transferring data.

The use of the Western Digital 1771 makes Southwest Tech's MF-68 one of the least software-dependent floppy-disk systems available to the hobbyist today. The highly effective data transfer rate, made possible by the use of the 1771, means this minifloppy disk is actually faster than some full-sized floppy-disk systems.

### So How Does It Work?

So far we have talked about the hardware and the software, but not the all-important end result. The two words that come to mind immediately when I think of the MF-68 are fast and smooth—fast because 12K of BASIC or CORES loads in a couple of seconds; smooth because the software provided with the MF-68, like all of the software written by Robert Uiterwyk, neither intrudes nor irritates—it just works. Come to think of it, what more could you ask from anything?

### Conclusion

Since I had the problem with the bad drive, I waited until the replacement arrived before finishing the assembly (i.e., mounting the front panel and cover on the drive enclosure). Much to my dismay, the holes did not line up on the front panel and the chassis. Why is it that someone can produce a disk system that works as well as the Southwest Tech MF-68 and fail to get the mounting holes for the front panel in the right places?■

*Robert M. Luby, Jr.*
*32 Westfield Rd.*
*Meriden CT 06450*

# The Coming Tragedy: Poorly Designed Small-Business Systems

We all know it: Computers are rapidly entering the small-business world. Hardware prices have nose-dived to the point where a small businessman can afford his own computer for accounting and bookkeeping. If you're a hobbyist, you probably have dreams of trying your hand at commercial systems. The market looks mouth-watering. Last year in the state of Connecticut alone, there were 1049 liquor stores, each a potential customer for a small system.

Sounds great, doesn't it? Yes, but watch out, there are questions to consider: "Who will be using the business system?"; "What requirements do the users have?"; and "How will those requirements be met?"

### Who'll Use the Business System?

Unlike a hobbyist's plaything, the small-business system is a tool, with but one function—making money. The businessman will use it, but others, namely, the owner's banker and the Internal Revenue Service, will also use it because they use the data that it generates. Moreover, they hold the businessman responsible for the correctness of the data he feeds them. Both expect the data they receive to be *what really happened*; they accept no excuses for anything less.

Generally, businessmen know little about computers. More important, they don't *want* to have to know anything about them. For example, one of those liquor-store owners is far more interested in selling booze, groceries, soup or his services than he is in computer repair or programming. Nevertheless, he remains responsible to his banker and the IRS, as well as to himself, for the correctness, accuracy and timeliness of his system's data.

How does a businessman serve his banker's and his government's need for data when he has purchased a computer? He delegates the responsibility to the system's designers and programmers. Although their customer is responsible for the data placed in the system, they must bear the responsibility for the data's integrity, accuracy and precision once it enters the system. Small-systems builders can meet these responsibilities by keeping systems' requirements in mind when they put the system together.

### Business System Requirements

The primary requirement in a business system is reliability, which must take precedence over the hobbyist's rightful interest in elegance, speed and technological advancement. Since the system will manipulate accounting data, it must meet accounting and auditing standards. If the system's owner does not demand this, his banker and the IRS certainly will.

A small system that maintains accounting data must satisfy two requirements: (1) the system must ensure that the data it manipulates is accurate, secure and reliable; (2) the system must organize the data so that an auditor can verify the accuracy, security and reliability of that data. When the data from any accounting system takes such a form, we have what accountants call an *audit trail*.

The audit trail is the chain of human-readable cross-references that allow an auditor to trace any figure produced by an accounting system back to the transactions that generated that figure. Without the audit trail, the data is untraceable, and its reliability, therefore, is unable to be proved. Omitting an audit trail from *any* accounting system is a grave error; in a small computer system, it is a *fatal* error. No system should ever change old data, or add new data without this audit trail, a pointer to the transaction causing the change.

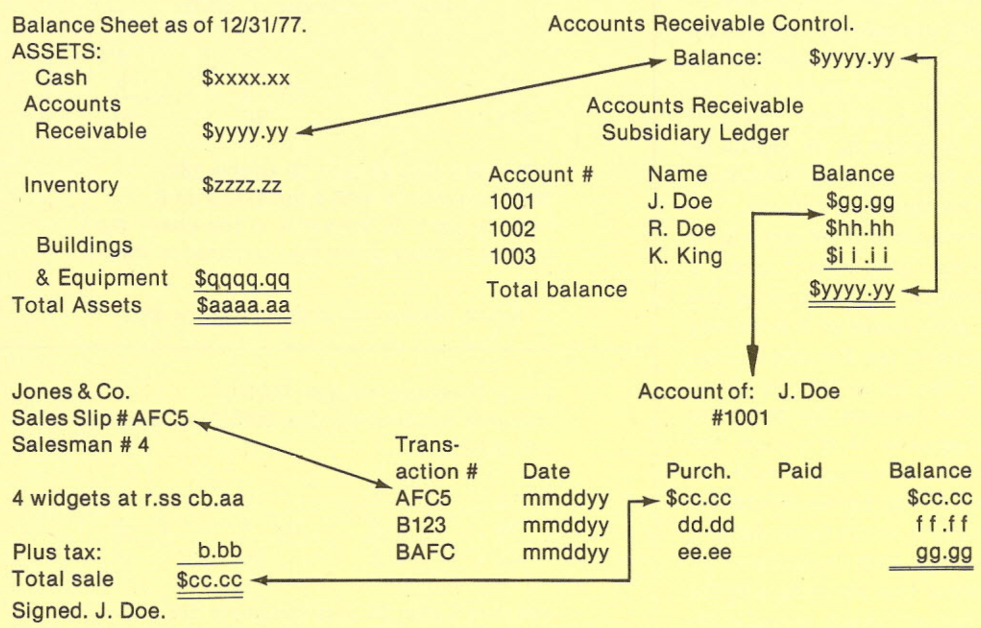
Fig. 1 illustrates how you can determine, through the audit trail, the total amount of money that customers owe a business (accounts receivable). For example, when you charge an item at a department store, the store keeps track of this in an accounts receivable set up under your name. Ultimately, the store's accounting system generates a summary figure that should be the total of the balances in the individual accounts. Since the vast majority of businesses and individuals pay their bills, the figure for receivables partially predicts that department store's future cash resources.

But wait! How do we know that no data has been lost in the summarization process? We don't know until we have periodically, on a sample basis, verified this total by tracing it back to individual transactions or otherwise inspecting the summarization process. Each entry in John Doe's account can be traced to either a sale or a payment. We can verify the system's mechanical accuracy simply by following the audit trail. After all, if we are to make good decisions about extending credit to Mr. Doe, we should have correct data with which to do it. Otherwise, it's garbage in-garbage out (GIGO).

### How to Meet These Requirements

The audit trail helps determine whether the system has been correctly manipulating accounting data. If the accounting data is inaccurate, it can help pin down the problem. The best system, though, is one that corrects errors by anticipating and preventing them. The best systems design, like the best medicine, is preventative. Like a disease, errors can infect any system severely enough to make it not only useless, but also dangerous. Any accounting system must control both the people using it and the data it processes if it is to be worthwhile.

A computerized system possesses two additional sources of error not contained in manual systems—the programs that control it and the hardware that




An accounting system is basically a tree structure of lists. The balance sheet is the list of the business's assets, liabilities and equities. Each number in the balance sheet should be backed up by one or more figures in the general ledger control accounts. One of these, accounts receivable, is illustrated here. These control accounts are backed by subsidiary ledgers containing the detailed list of activity for each account—in this illustration, J. Doe's. Each entry in Mr. Doe's account can be traced to a specific transaction that was recorded on a sales slip. The trunk of this tree is the balance sheet. One type of leaf is the sales slip.

*Fig. 1. The audit trail.*

embodies it. In a small system, these sources of error will have to be controlled by the systems builders themselves; there isn't anyone else to do it. The owner is too busy; his employees are interested in getting the job done as easily as possible. No one is left to control errors but the designers and programmers of the system, who must control four sources of error: people, programs, data and hardware.

**Controls over People**

People must be controlled for two reasons. First, people make mistakes; second, some people are larcenous. A regrettable truism in auditing circles is that the more trusted the employee, the larger the fraud he can perpetrate. In a small interactive system that gives rapid responses to most inquiries, the entire records of a business could be inspected, deleted or destroyed. Before they enter this field, erstwhile systems designers should give long and hard consideration to the piteous state of a business that has had its master files rendered unusable or unreliable. If they don't control these risks, they'll turn them into realities.

In designing a system you should make it easy for the owner to supervise his employees' use of the computer by catching errors before these errors contaminate vital data. Terminals can be locked. Files can be password-protected and enciphered. The system can be programmed to detect common errors and report their occurrence so they can be corrected. In particular, tasks can be divided among people so that either the owner participates in recording and verifying data or two employees check each other's work. This is called *separation of duties*. Individuals who control valuables should not keep the only records regarding them.

Designing error checks is particularly important on small systems, which are likely to be interactive, giving immediate responses to user commands—a seductive feature. Often no paper record of a transaction has to be printed, a great feature for playing "Hunt the Wumpus." The chief selling point of these systems is that the data they process need only be entered once—when the sale is made or the goods received. The burden of providing the edit checks that detect these errors rests squarely on the shoulders of the systems' builders.

Builders of systems should put modules into the system to correct errors gently and, above all, understandably. Programmers and engineers had better face the fact that a user confronted with error messages like "SN ERROR," "illegal operator" or, even worse, "Error 501," will be frustrated, unsettled and more likely to make mistakes. Requiring customers to remember strange abbreviations or to thumb through thick error manuals will only make things worse. The controls a system exerts over its users should not be so onerous that it inspires efforts to bypass it.

This doesn't sound easy, does it? It isn't. Nevertheless, if a small system is to work in the most basic sense, it must control the people who use it. Designers, engineers and programmers should make their systems easy to use and hard to abuse.

**Controls over Data**

Data should be computer-checked for accuracy. For example, most businesses purchase goods on credit and receive invoices, which should be checked for arithmetic errors when data is transferred to the computer. The system should look at an updated file of open purchase orders to determine whether the goods listed on the bill match the ordered goods. Amazingly enough, some people support themselves billing companies for merchandise that is unordered, as well as undelivered. A small system can catch this by cross-checking invoices with purchase orders. If such errors are made, the audit trail can pinpoint them.

The system should perform these accuracy and cross-reference checks before a master file is updated so that bad data will not contaminate it. Other pre-update checks should include:

1. A check that the part of the file about to be updated should be updated.
2. A series of checks to ensure that all new data is reasonable.
3. A check that the file change is authorized.
4. A check that all data are completely entered.
5. A check to ensure that the file change does not compromise the system's security.

When updating a record, every system should check to see whether it is indeed *this* record that should be updated. With thousands of John Smiths in the country, we need an account number to uniquely identify the particular John Smith who is liable on that account.

But numbers can be misread and people can transpose digits, which would send that data to the wrong account. One guard against this is the self-checking number, an account number with an extra digit appended. This digit is called a "check digit."

Fig. 2 shows how a check digit is generated by a popular check-digit system, the Mod-11 system. Fig. 3 shows how the same system detects a common error—the transposition of two digits in a number. A check digit system helps assure that data goes where it is supposed to go; that people's purchases are, in fact, charged to their own account.

In the Mod-11 check-digit system, each digit of the account number is multiplied by a different power of 2. These products are added, the sum divided by 11, and the remainder is subtracted from 11 to produce the check digit. This digit is then added to the end of the account number given the customer.

One edit step checks that the new data is reasonable. Certain input, obviously, makes no sense, such as a purchase of $Q3.86. (Somebody hit the Q key rather than the 2 key.) To guard against unusual data leading to outlandish results, data placed into a record section that describes a single attribute of a customer, called a field, should be compared with the character types allowed in the field. For example, a customer may owe money to a business. The business records this in the record devoted to that customer in the accounts-receivable file, which contains, for each customer, that customer's name, address, account number and open balance (what he owes the business). Decimal points, for example, should probably not appear in the name of a customer, nor should they appear in the account number. This type of check is called a *field check.*

Another category of edit checks that the new information falls within certain reasonable limits. This is the *limit check.* For example, in a small business where single sales of $1000 are unlikely, that figure could come up if someone hit a few too many zeros when he rang up a $10 sale. Errors of this type can be as painful as they are ridiculous. The following paragraph shows how.

A couple of years ago, a local Florida government used a computer to determine the property values so it could levy a property tax. The property-tax rate was set by dividing the required revenue by the total assessment. In this case, the person who keyed the data into the town computer moved the value of a house three digits to the left. This multiplied the value of that house by a factor of 1000, which was added to the total assessment. The resulting tax rate was too low, but by the time this error had been discovered, the tax rate had been set for that year. That local government found itself a few million dollars short. Had this system been programmed to flag all assessments over a certain amount as possible errors, it would have caught this error.

There are correct and incorrect ways to program error checks. Of course, good error checks take more effort to program than bad ones.

For example, suppose we wish to create a new record in a file. Part of this record will denote the sex of the subject. This would be useful in a clothing store's customer file. If the customer is male, the field will contain an M; if female, the field should contain an F. What would happen if this field contained an R? (Someone hit the wrong key again.) The first program fragment in Fig. 3a does not check for this "none of the above" situation. The second one does.

The reliable program specifies the desired response and checks for the required response before proceeding. The first program does not check for the "none of the above" situation that will occur, you can be sure. For example, the first program will take G for girl to mean male. If that type of error repeats, a girls' department

Account Number 9075.

| | | | |
|---|---|---|---|
| 9 | 0 | 7 | 5 |
| x 16 | x 8 | x 4 | x 2 |
| 144 | 0 | 28 | 10 |

144 plus 0 plus 28 plus 10 equals 182.
182/11 equals 16 with a remainder of 6.
11-6 equals 5. 5, therefore, is the check digit, and the number given the customer is 9075-5.

*Fig. 2. Generating a Mod-11 check digit.*

Correct account number: 9075-5.
Mistaken account number: 9057-5 (a transposition error).

| | | | |
|---|---|---|---|
| 9 | 0 | 5 | 7 |
| x 16 | x 8 | x 4 | x 2 |
| 144 | 0 | 20 | 14 |

144 plus 0 plus 20 plus 14 equals 178.
178/11 equals 16 with a remainder of 2. 11-2 equals 9. So the check digit should be 9, but a 5 has been entered. There must be a mistake in the account number, so the computer should now tell the operator about the error.

*Fig. 3. Detecting an error with a check digit.*

customer file will seem to consist largely of males with feminine names! If the business bases its marketing decisions on the large number of males who patronize it, those decisions will be wrong—and costly.

If something like this does happen, only a customer complaint will detect it. When the contaminated data is discovered, the audit trail of file changes will be crucial in identifying the problem and correcting the bad data that caused it, as well as the program that let this happen in the first place.

To track this down to a single employee, the files can be password-protected and user identities obtained by keeping passwords personally secret and by changing them at least monthly. Passwords, for example, should not be echoed on a terminal as the user keys one in because other people can see them. Passwords used to access files should be part of the audit trail as well as the changes made during program execution. Owners, in particular, should take password security seriously, for only then will they be able to impress this on their employees. Another access control is the terminal lock; still another is a time-of-day check to ensure that data is entered only during business hours.

On a more basic level, completeness checks see that all the blanks are filled in completely. To use my example of entering an account number, what would the system do if you entered the first two digits and then indicated that you were through? Would the system recover from that error? Remember, the smaller the business, the more likely these kinds of errors will occur. The owner will be depending on the computer to catch them.

To summarize, there is only one way to control the data in a computer system: check it, recheck it and check it again. In particular, check it before a file is updated so that good files will not be contaminated by bad data. When a file is contaminated, only the system's audit trail will be available to help in the reconstruction pro-

```
        Fragment I (unreliable)
200  PRINT "Please enter the sex of the customer."
205  INPUT Sex
210  IF Sex is equal to "F" perform Female-Procedure and then GO TO Rest-of-Program.
220  Perform Male-Procedure.
230  Label: Rest-of-program.
     the rest of the program . . . . . . . . . . . .

        Fragment II (more reliable)
200  PRINT "Please enter the sex of the customer; either "M" or "F".
205  INPUT Sex.
210  IF Sex is equal to 'F' perform Female-Procedure and then GO TO Rest-of-Program.
220  IF Sex is equal to 'M' perform Male-Procedure and then GO TO Rest-of-Program.
230  Perform Sex-error Procedure and then GO TO 200.
240  Label: Rest-of-Program.
     the rest of the program . . . . . . . . . . . .
```

*Fig. 3a. Error-check examples.*

cess. The system must maintain a comprehensive, permanent audit trail to assure recovery from the inevitable errors. Limit checks, reasonableness checks, accuracy checks, authorization checks, completeness checks, check digits and field checks control errors and maintain the purity of the data the system maintains. This purity is the system's most precious component. The owner will depend on the system's designers, acting through the computer, to protect it.

The system's data should also be secure from fire and theft. Master files should be copied on a regular basis and then stored off the premises in a safe place.

**Controls over Programs**

Programs come in two varieties—applications programs, which instruct the system to perform some useful task, and systems programs, which control the system's operations in performing that task. For example, an applications program would control the system so that the computer would print mailing lists, while the operation of the Teletype used to print the list would be controlled by a systems program used by other applications programs, as well as the mailing-list-printing program.

You can control applications programs—edit all inputs, provide adequate error messages and test the program thoroughly. Given omnipresent Murphy's law, the worst errors will probably happen at the worst possible time (i.e., when your customer runs his payroll). Hardware problems can be excused, but it's difficult to explain to your customer why his books won't be closed on time because you can't find the error in *your* program.

Another "control" is to understand the business! Only then can you start flowcharting and writing the program. There is a great deal to learn—for example, inventory systems.

One product of an inventory system is a figure for total inventory. This figure can be computed in any of the following ways: (1) first in-first out (FIFO); (2) last in-first out; (3) specific identification; (4) lower of cost or market; (5) replacement cost; (6) retail value method.

These are a few of the more

popular ways to compute that figure for total inventory, which seemed so simple a few paragraphs ago. To complicate matters further, your customer may choose to value parts of his inventory by different methods. An auto dealer, for example, might use specific identification to value his inventory of cars, but use FIFO to value his parts inventory. Clearly, then, the hobbyist looking for extra money by selling inventory systems had better know what he's doing before pencil is set to paper. If these things aren't considered, the programmer will not only imperil his client, he'll impoverish him.

Another threat to the system's reliability comes from the systems programs that control its fundamental functions. If numbers are recorded with insufficient precision, round-off errors will pile up. As a case in point, the six decimal digits provided by some BASIC interpreters are not adequate for a business application. Assuming that a small businessman wants to clear at least $10,000 to support his family, we can determine how large this figure must be. If he has a five percent profit margin on his sales, they will have to be $200,000.00.

This is an eight-digit number (oops!) and we have not included his liabilities and stake in the business, which are added to his sales to see whether his books balance. To fit even the smallest business, then, the system should provide at least eight digits of precision, with ten to twelve preferred. Adding numbers in this form is slower, but the trade-off is acceptable.

To protect systems programs from inadvertent modification, they can be placed in ROM (read-only memory). This also makes them harder to copy, since anybody who wished to pirate the program would have to interpret a listing in hexadecimal, or an uncommented listing produced by a disassembler program. They'd better be debugged before they are placed in ROM, though.

**Control of Hardware**

After my dismal recitation of the myriad miseries that can infect a small system's data and programs, I must mention another concern. Hardware should perform reliably. Tactics include read-after-write, checksums, error-correcting codes and parity checks. Yet most S-100 bus memory boards do not include parity bits. In a business system, that is a false economy. If a bit flips in a game system, the worst that can happen is that you will have to start over—a minor inconvenience. In a business environment, the same error could mean lost time, at least—at most, lost customers.

The system's audit trail must be made absolutely reliable. The tape drive or cassette deck, the floppy disk or printer, must preserve this data. Recovery from inevitable errors will be impossible without it. If the audit trail is not preserved, the businessman may have to hire a bookkeeper to reconstruct his records, which could take months. I previously mentioned that a businessman purchases a computer to make money. He does not make money by hiring a bookkeeper for several months because an incompetently designed computer system destroyed its data or lacked an audit trail to reconstruct it.

**Summary**

Business users, above all, demand reliability from their computers. Therefore, business systems must be designed with control and verifiability built in. Behind their design they must have a philosophy that values control and reliability more highly than these factors are valued in hobbyist systems. Consequently, many hobbyist-level systems will probably prove inadequate in a business environment.

The best way to correct these deficiencies is to prevent them with controls. If you're entering this field, help your client control people using the system, the data it processes, the hardware that embodies it and the programs that control it. Control is your goal. I've shown a few ways to reach it. ■

*New!* **SELECTERM** SYSTEM 9710

# The SELECTRIC II* Printer you can TRUST

**BECAUSE . . .** It's brand new, and fully assembled and tested.

**BECAUSE . . .** After extensive engineering design and testing by Micro Computer Devices, IBM Corporation has approved the SELECTERM for use with your microcomputer, and provides you with their factory warranty and yearly service agreement for the typewriter. In addition, the electronics conversion portion is fully warranteed by Micro Computer Devices.

**BECAUSE . . .** You can connect the SELECTERM to your computer within minutes of taking it out of the carton.

IT'S THAT EASY! AND THAT RELIABLE!

**FEATURES**
- Complete ASCII character set in supplied element.
- Full upper, lower case alphanumeric characters.
- Tab Command, Index (vertical tab), Backspace, Bell—all under computer control.
- Parallel Interface, standard.

**ALL ELECTRONICS INCLUDED**
- Power supply, electronics and cable sets included to permit immediate connection to the parallel port of any computer, at standard TTL level.

**SOFTWARE**
- All necessary conversion software in PROM to handle ASCII input, directly.

**PRINTER or TYPEWRITER**
- May be used as a standard typewriter when not in use with your computer.

**OPTIONS**
- Dual Pitch, $125
- Correction Feature, $125
- Tractor Feed Platen, $250
- Noise Reduction Feature, $50

**AVAILABLE SOON**
- RS-232 Interface

**PRICE and DELIVERY**
- Assembled and tested, $1750
- Available ONLY from authorized dealers.
- Delivery 1 to 2 weeks from receipt of order.
- OEM delivery in quantity within 30 days.

SELECTERM

micro computer devices inc.
960 E. Orangethorpe, Bldg. F
Anaheim, California 92801
Telephone (714) 992-2270

*Registered trademark of IBM Corporation

M30

*"Innovators to the Microcomputer Industry"*

James P. Fentress

# Utility Routines

## *useful programs for your 6800*

As a fellow microcomputer addict, you have undoubtedly experienced a need for more and better software. Surely there have been times when having a useful utility routine available would have saved you a lot of programming and debugging time. However, even if you could have looked through your past magazine issues for a routine to do the job, the possibility of easily adapting that routine to fit your system would be very small. I am suggesting – and offering – a standard routine interface format to simplify the task of exchanging assembly language routines for the 6800 microprocessor.

The interface program is called MONTOR, and is a 6800 implementation of the executive routine presented by Dick Wilcox in the February, 1977 issue of *Kilobaud*. (Please refer to his article for a good explanation of an executive, or monitor call processor.) The executive performs all the necessary housekeeping chores to make the system work, and now routines can be added to your library with a minimum of hassle or intensive debugging. It is not necessary to be a heavy programmer to add a routine to your library; just follow a few simple rules when calling a routine and you (and the routine) are in business.

I have included with MONTOR a couple of utility routines that I feel can be useful to any operating system. These routines are primarily given as examples of how the routines might be written for the MONTOR library, but the neat thing about the system is that if they are left out, the system operation is not altered in any way. You just include those routines of interest to you and your system.

The two routines, EBCASC and ASCEBC, are code conversion routines for translating EBCDIC characters to ASCII and vice versa. Those systems with an EBCDIC-encoded keyboard and an ASCII-coded display would find the EBCASC routine very useful (I've seen a lot of beautiful surplus EBCDIC keyboards sold for less than $20, so this setup seems to be a likely possibility). Certain ASCII-coded systems output to EBCDIC-coded displays, or to larger computer systems which are EBCDIC character-code based. If your system is one such as this, the ASCEBC routine can be very useful to you. Other routines in the works are: preset memory, move multiple memory blocks, and multiple block tape output. I'll discuss these and other routines in more detail later in this article.

Some considerations have been incorporated in the routines presented in order to make the monitor usable in a wide variety of system configurations. First, the monitor routines had to be reentrant so they could be used in a real-time environment. System interrupt processors should have access to the monitor library as well as regular programs. Making the routines reentrant prevents some of the problems encountered when putting together a multiple-level operating system.

Some interesting extras of these reentrant routines are the possibility of writing re-

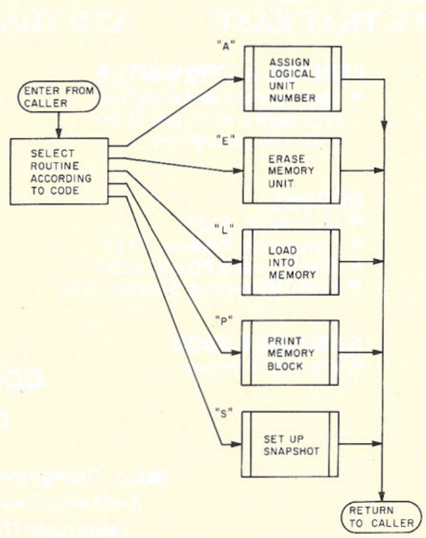
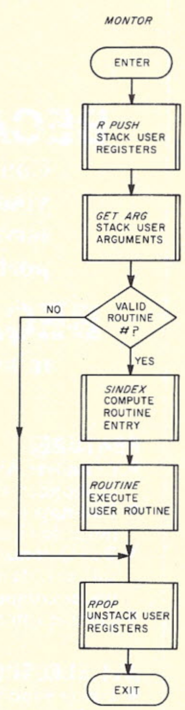



*Fig. 1. Monitor example.*




**MONTOR** – The MONTOR flow diagram shows a series of subroutine calls, and only one decision made: Is the routine number a valid routine, that is, did the caller specify a subroutine that MONTOR can't possibly have? (Routine #>128).

*Fig. 2. MONTOR.*

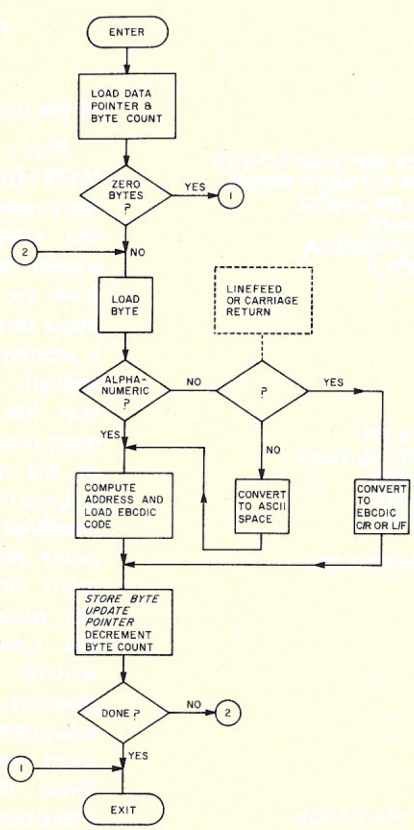
cursive library routines, writing routines that call other library routines, and operating library routines from read-only memory. (Recursive programming is really a unique experience if you like programming. Rather than make a poor attempt at explaining it, I am directing interested readers to programming texts such as *Programming Languages: Design and Implementation* by Terrence W. Pratt, Prentice Hall Publications, for more information on implementing recursive routines.)

Some of the other considerations made in writing the software were that the overhead for calling a monitor library routine be kept to a minimum, that the subroutine calling format allow flexibility in passing and receiving information to and from the different routines, and that there be as little difficulty as possible in getting each new addition to the library up and running. (It shouldn't be necessary to carefully analyze each new routine in order to add it to your system!) If you don't readily see the logic in these considerations, please refer to Dick Wilcox's February, 1977 *Kilobaud* article for some good general information on monitor call processors. Especially useful is his explanation of the *raison d'etre* of reentrant routines.

## MONTOR Operations

I'm a simplistic programmer, and write programs in as straightforward a manner as possible when solving a problem. As can be seen from the flowcharts, MONTOR is no exception. The monitor call processor must perform certain operations, most of which are outlined in the MONTOR flow diagram. The first operation is to save the contents of the caller's registers on the stack, where they can be kept until the library routine returns control. If the register contents were saved in some absolute memory location, the caller's registers would be overwritten if the library routine called another library routine via MONTOR. By saving them on the stack instead, routine calls can be nested to a depth limited only by the size of the stack area. The routine RPUSH accomplishes the actual register-push-to-stack operation (again in a very straightforward manner, as can be seen from the RPUSH flow diagram), and the actual order of placing the registers on the stack is given with the program listing for RPUSH.

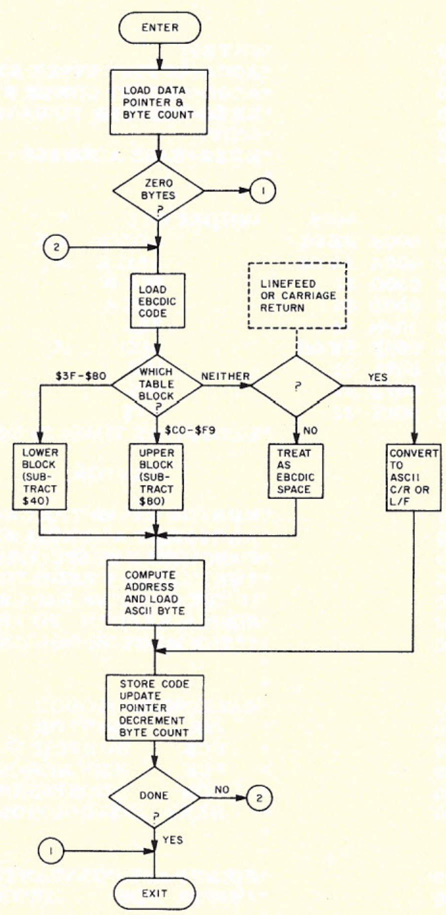
*ASCEBC*




**ASCEBC** — Although this flowchart may look complicated, there are only four decisions being made. This routine operates by means of an indexed table: hexadecimal $20 is subtracted from the ASCII character, then the result is added to the base address of the conversion table. The decisions made are: 1. Did the caller specify a zero byte count? (No operation.) 2. Is the current byte a number or character? 3. Is the current byte a carriage return or line feed? 4. Are we done yet? Some points should be mentioned here — lowercase letters and control codes are *not* decoded. This cuts down considerably on the size of the conversion table. If a code is not converted, it is changed to an EBCDIC space code. The routine is *fast*, as conversion routines go.

*Fig. 3. ASCEBC.*

*EBCASC*




**EBCASC** — EBCASC is basically the same routine as ASCEBC, with the prime difference being that the conversion table has been split into two blocks. This split is necessary to conserve table space, because the EBCDIC character codes are not so nicely consecutive as are the ASCII codes. The only extra decision made in this routine is to find out which table block the character to be converted lies in. Depending on the result of this decision, a different quantity is subtracted from the character code, but then everything else is the same. The extra decision costs about 20 usec timewise, and several extra bytes of code. Note that only uppercase letters are converted, as with the ASCII-to-EBCDIC routine (ASCEBC).

*Fig. 4. EBCASC.*

Since I've now referred to the program listing, I should explain some of the conventions used in commenting the listing:

1. ACCA, ACCB refer to the A and B accumulators, respectively.

2. XREG refers to the X, or index register.

3. CC refers to the condition codes register.

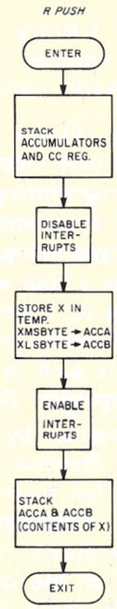
4. ADX is an abbreviation for address.

5. HI, LO refer to the upper 8 bits and the lower 8 bits of a 16 bit value respectively.

6. The contents of the stack pointer (S.P.) are listed from the most recent entry (at the top) down to the least recent entry (at the bottom). This makes the stack's memory address order an ascending series (as a program listing always is) with the lowest address at the top and the highest address at the bottom of the listing.

*Program A. Source Listings for MONTOR.*

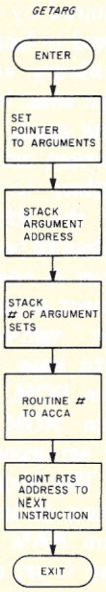
```
               *FIRST PAGE CONSTANTS AND VARIABLES
0000            XSAV   RMB     2          TWO BYTE TEMP STORAGE FOR
               *                  INDEX COMPUTATIONS WITH INTERRUPTS OFF
0002 01F5       EANDX  FDB     EATAB      ADX OF EBCDIC TO ASCII TABLE
     0000       AEXSAV EQU     XSAV       INDEX COMPUTATION TEMP WORD
0004 026F       AENDX  FDB     AETAB      ADX OF ASCII TO EBCDIC TABLE
     0000       EAXSAV EQU     XSAV       INDEX COMPUTATION TEMP WORD
0006 02AF       JMPPTR FDB     JMPTAB     ADDRESS OF LIBRARY ROUTINE TABLE
               *
               *
               *DINDX2: DOUBLE PRECISION ADD OFFSET TO BASE ADX
               *DINDX2 IS NOT USED FOR THIS VERSION OF
               *MONTOR BUT IS AVAILABLE FOR EXPANSION TO 32K ROUTINES.
               *
               *
               *ENTRY:
               *ACCA=OFFSET UPPER BYTE
               *ACCB=OFFSET LOWER BYTE
               *XREG=POINTER TO BASE ADDRESS UPPER BYTE
               *EXIT:
               *XREG=BASE ADDRESS + OFFSET
               *
               *
     0008       DINDX2 EQU     *
0008 EB 01             ADD B   1,X        ADD OFFSET LOWER TO BASE LOWER
000A A9 00             ADC A   X          ADD OFFSET UPPER TO BASE UPPER
000C 37                PSH B              PUSH LOWER BYTE ON STACK
000D 36                PSH A              PUSH UPPER ON STACK
000E 30                TSX                X POINTS TO RESULT LOWER
000F EE 00             LDX     X          LOAD RESULTS INTO X
0011 31                INS                RESTORE STACK
0012 31                INS
0013 39                RTS                RETURN
               *EXECUTION TIME= 39 USEC

                          MONTOR

               *MONTOR SAVES THE CALLER'S REGISTERS,SAVES THE
               *ARGUMENT ADDRESS AND NUMBER OF ARGUMENTS, THEN
               *EXECUTES THE SPECIFIED ROUTINE AND RESTORES
               *THE CALLER'S REGISTERS AND RETURNS CONTROL.
               *IF THE ROUTINE #IS GREATER THAN 127, MONTOR
               *SIMPLY RETURNS TO THE CALLER.
               ***THE MONTOR ROUTINES OCCUPY LESS THAN 100 BYTES
               *
               *
               *CALLING PROTOCOL:
               *   JSR     MONTOR
               *   FCB     ROUTINE #
               *   FCB     #OF ARGUMENTS
               *   FDB     ADDRESS OF FIRST ARGUMENT
               *   NEXT  INSTRUCTION
               *
               *
               *FIRST PAGE CONSTANTS:
               *JMPPTR FDB     JMPTAB     *POINTER TO ROUTINE TABLE
               *
               *
0100                   ORG     $100       ARBITRARY STARTING ADX OF MONTOR
     0100       MONTOR EQU     *
0100 8D 1F             BSR     RPUSH      SAVE CALLER'S REGISTERS
0102 8D 40             BSR     GETARG     STACK ARGUMENTS,GET RTN #
0104 4D                TST A              SEE IF ARGUMENT OUT OF RANGE
0105 2B 0A             BMI     MONRET     DO NOTHING IF OUT OF RANGE
0107 48                ASL A              CHANGE ROUTINE #TO 2 BYTE VALUE
0108 CE 0006           LDX     #JMPPTR    POINT X TO ROUTINE TABLE ADX
010B 8D 07             BSR     SINDEX     COMPUTE ROUTINE ADDRESS
010D EE 00             LDX     X          PUT ROUTINE ADX IN X REG
010F AD00              JSR     X          EXECUTE SPECIFIED ROUTINE
0111 8D 21      MONRET BSR     RPOP       RECALL CALLER'S REGISTERS
0113 39                RTS                AND RETURN CONTROL
               *EXECUTION TIME APPROX 40 USEC (MONTOR CODE)
```




**RPUSH** — RPUSH serves only to save the caller's registers on the stack. Care is taken not to perform any operations that would change any program flags before they are set on the stack.

*Fig. 5. RPUSH.*

Referring back to MONTOR operations, the next operation after stacking the caller's registers is to obtain the argument pointers from the calling program and place them on the stack. This is accomplished by GETARG, logically enough, and in addition the caller's return program counter (that was saved on the stack by the jump to subroutine instruction) is modified by GETARG to point past the caller's argument pointers. If this were not done then returning to the caller from MONTOR would produce the unpleasant effect of executing a subroutine number, an argument count, and an address. Since the results of this execution can be difficult to predict, I decided to avoid the situation altogether. (Refer to the GETARG flowchart for another example of straightforward programming.) The listing of GETARG shows the stack contents before and after execution, and the on-exit format is the stack format that each library routine




**GETARG** – GETARG is an excellent example of straightforward programming. The pointers to the caller's argument set are placed on the stack, the routine number is in accumulator A, the program counter (on the stack from the "JSR" to MONTOR) is modified, and GETARG returns.

*Fig. 6. GETARG.*

assumes in order to access the caller's arguments. An explanation of the format (Example 1) is in order at this point.

\# OF ARGUMENTS IN LIST: This is a one-byte value that specifies the number of argument sets being passed to the library routine by the caller. It allows a library routine to process more than a single operation at a time and serves to reduce the overhead of the monitor call processor to a minimum by allowing the caller to save up as many jobs as possible then do them all at once. For example, a multiple block punch routine can output as many blocks of memory as desired (up to 256). Since each block can be any size, and different blocks need not be in any specific order in memory, it is possible to be very flexible in outputting memory contents to the system storage device.

ARG LIST PTR: This is a two-byte address pointing to the first argument set to be used by the library routine. The argument set can be any-

*SINDEX*

```
                *SINDEX ADDS THE CONTENTS OF ACCA TO
                *THE 16 BIT VALUE POINTED TO BY THE X
                *REGISTER, AND RETURNS THE RESULT IN X.
                *
                *
                *ENTRY:   ACCA=OFFSET TO BE ADDED TO 16 BIT VALUE
                *         XREG POINTS TO THE 16 BIT VALUE
                *EXIT:  XREG=OFFSET + 16 BIT VALUE
                *
                *
          0114   SINDEX EQU     *
0114 5F                 CLR B              ZERO VALUE OF UPPER OFFSET BYTE
0115 AB01               ADD A   1,X        ADD OFFSET TO LOWER BYTE
0117 E9 00              ADC B   X          ADD IN CARRY BIT TO UPPER BYTE
0119 36                 PSH A              PUT LOWER BYTE ON STACK
011A 37                 PSH B              PUT UPPER BYTE ON STACK
011B 30                 TSX                POINT X TO STACKED VALUE
011C EE 00              LDX     X          LOAD OFFSET + VALUE
011E 31                 INS                RESTORE STACK POINTER
011F 31                 INS
0120 39                 RTS                AND RETURN
                *EXECUTION TIME APPROX 47 USEC.
```

*RPUSH*

```
                *RPUSH SAVES THE MONTOR CALLER'S REGISTERS.
                *STACK ORDER:  XREG HI BYTE
                *              XREG LO BYTE
                *              CC
                *              ACCB
                *              ACCA
                *
                *
                *FIRST PAGE REQUIREMENTS:
                *XSAV      RMB 2    (TEMP STORAGE FOR X REGISTER)
                *
                *
          0121   RPUSH  EQU     *
0121 36                 PSH A              STACK ACCA
0122 37                 PSH B              AND ACCB
0123 07                 TPA                GET CC REG
0124 36                 PSH A              AND STACK IT
0125 0F                 SEI                DISABLE INTERRUPTS
0126 DF 00              STX     XSAV       SAVE CALLER'S X REG
0128 96 00              LDA A   XSAV       GET X HI
012A D6 01              LDA B   XSAV+1     GET X LO
012C 0E                 CLI                REENABLE INTERRUPTS
012D 37                 PSH B              STACK X LO
012E 36                 PSH A              STACK X HI
012F 30                 TSX                POINT TO STACK
0130 EE 05              LDX     5,X        PUT RETURN ADX INTO X REG
0132 6E 00              JMP     X          AND EFFECT A RETURN FROM SUBROUTINE
                *EXECUTION TIME APPROX 56 USEC
```

*RPOP*

```
                *RPOP RESTORES THE MONTOR CALLER'S REGISTERS.
                *THE EXPECTED STACK ORDER IS:
                *   MONTOR RTRN ADX HI
                *   MONTOR RTRN ADX LO
                *   XREG HI
                *   XREG LO
                *   CC
                *   ACCB
                *   ACCA
                *
                *
          0134   RPOP   EQU     *
0134 32                 PUL A              GET RTRN ADX HI OFF STACK
0135 33                 PUL B              AND LO
0136 30                 TSX                POINT INDEX TO STACK
0137 A7 05              STA A   5,X        PUT RTRN ADX HI INTO POSITION
0139 E7 06              STA B   6,X        AND THE ADX LO FOR THE RTS
013B EE 00              LDX     X          GET X REG OFF STACK
013D 31                 INS                MOVE STACK POINTER
013E 31                 INS                TO CORRECT POSITION
013F 32                 PUL A              UNSTACK CC CONTENTS
0140 06                 TAP                RESTORE CC
0141 33                 PUL B              RESTORE ACCB
0142 32                 PUL A              RESTORE ACCA
0143 39                 RTS                AND RETURN
                *EXECUTION TIME APPROX 52 USEC
```

*GETARG*

```
                *GETARG STACKS MONTOR CALLER'S ARGUMENT POINTERS
```

```
                    *THEN SETS ACCA TO THE ROUTINE#.
                    *
                    *
                    *ENTRY:   STACK =
                    *   MONTOR RTRN ADX HI
                    *   MONTOR RTRN ADX LO
                    *   XREG HI
                    *   XREG LO
                    *   CC
                    *   ACCB
                    *   ACCA
                    *   ARGUMENT ADX HI
                    *   ARGUMENT ADX LO
                    *
                    *
                    *EXIT:   STACK =
                    *   #OF ARGUMENTS IN ARG LIST
                    *   ARG LIST PTR HI
                    *   ARG LIST PTR LO
                    *   MONTOR ADX HI
                    *   MONTOR ADX LO
                    *   XREG HI
                    *   XREG LO
                    *   CC
                    *   ACCB
                    *   ACCA
                    *   CALLER RTRN ADX HI
                    *   CALLER RTRN ADX LO
                    *
                    *
                    *ACCA CONTAINS ROUTINE #
                    *
                    *
      0144          GETARG EQU    *
0144  30                   TSX               SET POINTER TO STACK
0145  EE 09                LDX    9,X        ARG POINTER TO X REG
0147  A6 02                LDA A  2,X        ARG ADX HI TO ACCA
0149  E6 03                LDA B  3,X        LO TO ACCB
014B  37                   PSH B             STACK LO
014C  36                   PSH A             STACK HI
014D  E6 01                LDA B  1,X        #ARGS TO ACCB
014F  37                   PSH B             AND STACK IT
0150  A6 00                LDA A  X          SUBROUTINE #TO ACCA
0152  30                   TSX               X REG POINTS TO #ARGS ON STACK
0153  C6 04                LDA B  #4         UPDATE CALR ADX LO
0155  EB 0D                ADD B  13,X       ADD 4 TO CALR RETURN ADX
0157  E7 0D                STA B  13,X       AND RESTORE RESULTS
0159  24 02                BCC    *+4        IF NO CARRY, RETURN
015B  6C 0C                INC    12,X       ELSE PROPAGATE CARRY TO UPPER
015D  EE 03                LDX    3,X        AND EFFECT A RETURN
015F  6E 00                JMP    X          FROM SUBROUTINE (RTS)
                    *EXECUTION TIME APPROX 70 USEC
```

*ASCEBC*

```
                    *ASCEBC CONVERTS AN ASCII BUFFER TO EBCDIC
                    *CAPITAL LETTERS ONLY ARE CONVERTED
                    *
                    *
                    *ENTRY:   STACK =
                    *   MONTOR ADX HI    (REFERENCE ONLY)
                    *   MONTOR ADX LO
                    *   #ARGS IN LIST (DUMMY, NOT USED)
                    *   ARG LIST ADX HI
                    *   ARG LIST ADX LO
                    *   MONTOR RTRN ADX HI ( USED BY RTS)
                    *   MONTOR RTRN ADX LO
                    *
                    *
                    *AT ARG ADX  CALLER PROTOCOL IS:
                    *   BUFFER HI ADX
                    *   BUFFER LO ADX
                    *   #BYTES IN BUFFER
                    *
                    *
                    *EXIT:  SPECIFIED BUFFER IS NOW EBCDIC
                    *   NOTE:  MAX BUFFER SIZE IS 256 (COUNT = 255)
                    *BASE PAGE REQUIREMENTS:
                    *AEXSAV RMB      2
                    *AENDX  FDB      AETAB
                    *
                    *
      0161          ASCEBC EQU    *
0161  30                   TSX               SET POINTER TO STACK ARG'S
0162  33                   PUL B             GET RETURN ADX HI
0163  32                   PUL A             GET RET ADX LO
0164  A7 06                STA A  6,X        SET LO ON STACK
0166  E7 05                STA B  5,X        AND HI
0168  EE 03                LDX    3,X        GET ARG POINTER
016A  33                   PUL B             PULL DUMMY OFF STACK
```

thing required by the library routine, and in fact is defined by the library routine depending on what information the routine needs or returns. The argument set might be two numbers to be multiplied, or the address of a block of data to be processed, or whatever is needed. Since the argument set can be anywhere in memory, it is possible to have the calling program in ROM, the library routine in ROM, and the argument set somewhere off in RAM. The library routine doesn't have to know where the system RAM is, or how the system is set up — all it needs to know is where to find the argument set, and that information is on the stack in the form of the ARG LIST PTR. Incidentally, the argument set can be bi-directional, passing information from the library routine to the caller as well.

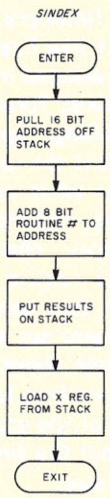
The routine GETARG gets one last unit of information from the caller: the routine number. The library routines are effectively an ordered table beginning with routine #0 up to routine #127. GETARG picks out the routine number and leaves it in accumulator A, then returns to MONTOR.

MONTOR's next operation is a check of the routine number. Since it won't do to try executing a routine that isn't available, the subroutine number is checked for a value greater than the maximum possible number of routines, in this case 127. If the routine number is out of range, control is simply returned to the caller. When the routine number is in range, the starting address of the desired library routine is computed by SINDEX.

The starting addresses of the library routines are kept in a table called JMPTAB, and when SINDEX computes the address of the requested routine, the routine number is multiplied by two (because each address is two bytes long) and added to the base address of the jump table, JMPTAB. This resulting address is returned in the index (X) register by SINDEX for use by MONTOR.

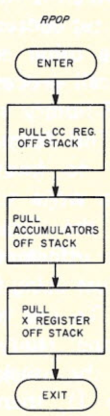
MONTOR executes an indexed jump to the subroutine, which effectively passes control to the library routine. When the library routine finishes processing all requested operations, it returns control to MONTOR.

Since everything is done at this point, MONTOR reloads




**SINDEX** — What could be simpler? SINDEX adds an eight bit offset to the address of the jump table "JMPTAB," leaves the result in the X register, and returns.

*Fig. 7. SINDEX.*




**RPOP** — RPOP recalls the previously saved caller's registers from the stack, and returns. The rather cryptic name is an abbreviation for "register pop," which is exactly what this routine does.

*Fig. 8. RPOP.*

```
2180 016B E6 02          LDA B  2,X        GET BYTE COUNT
2190 016D 27 1E          BEQ    AERET      RETURN IF ZERO BYTES
2200 016F EE 00          LDX    X          DATA POINTER TO X REG
2210 0171 A6 00   AELOOP LDA A  X          GET ASCII CODE
2220 0173 2A 02          BPL    ASCOK      IF GOOD DATA, CONVERT TO EBCDIC
2230 0175 86 20          LDA A  #$20       ELSE TREAT AS SPACE
2240 0177 80 20   ASCOK  SUB A  #$20       SET UP INDEX INTO TABLE
2250 0179 2B 15          BMI    CRLFCK     BRANCH TO CHECK CAR RET OR LINE FEED
2260 017B 0F             SEI               DISABLE INTERRUPTS
2270 017C DF 00          STX    AEXSAV     SAVE DATA POINTER
2280 017E 97 05          STA A  AENDX+1    SET UP CONVERSION INDEX
2290 0180 DE 04          LDX    AENDX      INDEX TO X REG
2300             *ERROR 15 IS EXPECTED AND ALLOWED IN THE NEXT INST.
2310             *THE UPPER 8 BITS OF THE ADX ARE TRUNCATED
ER15 line 2320
2320 0182 A6 6F          LDA A  AETAB,X    ADD LOWER 8 BITS OF TAB ADX TO X
2330             *NOW THE CONVERTED DATA IS IN ACCA
2340 0184 DE 00          LDX    AEXSAV     RELOAD DATA POINTER
2350 0186 0E             CLI               REENABLE INTERRUPTS
2360 0187 A7 00   STORE  STA A  X          STORE CONVERTED DATA
2370 0189 08             INX               UPDATE POINTER
2380 018A 5A             DEC B             DECREMENT BYTE COUNT
2390 018B 26 E4          BNE    AELOOP     LOOP UNTIL DONE
2400 018D 31      AERET  INS               RESTORE STACK POINTER
2410 018E 31             INS
2420 018F 39             RTS               AND RETURN
2440 0190 8B 20   CRLFCK ADD A  #$20       RESTORE ORIGINAL CODE
2450 0192 81 0A          CMP A  #10        IS IT A CARRIAGE RETURN?
2460 0194 26 04          BNE    LFCK       IF NOT CHECK FOR LINE FEED
2470 0196 86 0D          LDA A  #$0D       EBCDIC CARRIAGE RETURN
2480 0198 20 ED          BRA    STORE
2490 019A 81 0D   LFCK   CMP A  #13        IS IT A LINE FEED
2500 019C 26 D7          BNE    ASCOK-2    IF NOT PROCESS AS SPACE
2510 019E 86 25          LDA A  #$25       ELSE CONVERT TO EBCDIC LINE FEED
2520 01A0 20 E5          BRA    STORE
2530             *EXECUTION TIME APPROX 80 USEC
2540             *LOOP TIME APPROX 60 USEC PER CONVERSION
```

*EBCASC*

```
2560             *EBCASC CONVERTS AN EBCDIC BUFFER TO ASCII
2561             *CAPITAL LETTERS ONLY ARE CONVERTED
                 *
                 *
2570             *ENTRY:  STACK =
2580             *   MONTOR RTN ADX HI (USED FOR REFERENCE)
2590             *   MONTOR RTRN ADX LO
2600             *   #OF ARGS IN LIST  (DUMMY, NOT USED)
2610             *   ARG LIST ADX HI
2620             *   ARG LIST ADX LO
2630             *   MONTOR RTRN ADX HI  (USED FOR RTS)
2640             *   MONTOR RTRN ADX LO
                 *
                 *
2650             *AT ARG LIST ADX, CALLER PROTOCOL IS:
2660             *   BUFFER ADX HI
2670             *   BUFFER ADX LO
2680             *   #OF BYTES TO BE CONVERTED
                 *
                 *
2700             *EXIT:  EBCDIC BLOCK IS CONVERTED TO ASCII
2710             *BASE PAGE REQUIREMENTS:
2720             *EAXSAV RMB     2
2730             *EANDX  FDB     EATAB
                 *
                 *
2750      01A2    EBCASC EQU    *
2760 01A2 30             TSX               SET POINTER TO ARG ADX
2770 01A3 33             PUL B             RTRN ADX HI OFF STACK
2780 01A4 32             PUL A             GET LO OFF STACK
2790 01A5 A7 06          STA A  6,X        PUT RTRN ADX ON STACK
2800 01A7 E7 05          STA B  5,X        AND ADX HI
2810 01A9 33             PUL B             PULL DUMMY OFF STACK
2820 01AA EE 03          LDX    3,X        LOAD ARG ADX
2830 01AC E6 02          LDA B  2,X        GET BYTE COUNT INTO ACCB
2840 01AE 27 42          BEQ    EARET      IF ZERO RETURN
2850 01B0 EE 00          LDX    X          LOAD DATA POINTER
2860 01B2 A6 00   EALOOP LDA A  X          GET DATA BYTE
2870 01B4 81 C0          CMP A  #$C0       IS DATA IN UPPER TABLE BLOCK?
2880 01B6 24 22          BCC    UPRBLK     IF SO, DO IT
2890 01B8 81 80          CMP A  #$80       ELSE CHECK IF IN LOWER BLOCK
2900 01BA 25 16          BCS    LWRBLK     AND PROCESS IF IN TABLE
2910 01BC 81 0D   EAERR  CMP A  #$0D       CHECK IF EBCDIC CAR RET
2920 01BE 26 04          BNE    ALFCK      IF NOT CHECK FOR LINE FEED
2930 01C0 86 0A          LDA A  #10        CONVERT TO ASCII CAR RET
2940 01C2 20 28          BRA    STORA
2950 01C4 81 25   ALFCK  CMP A  #$25       IS IT A LINE FEED?
2960 01C6 26 04          BNE    MKSPC      IF NOT,TREAT AS SPACE
2970 01C8 86 0D          LDA A  #13        ELSE CONVERT TO ASCII LINE FEED
2980 01CA 20 20          BRA    STORA      RESTORE IT
```

```
01CC 86 20   MKSPC  LDA A  #$20      CONVERT TO ASCII SPACE
01CE 20 1C          BRA    STORA
01D0 20 0E          BRA    LOAD      AND CONVERT TO ASCII
01D2 81 3F   LWRBLK CMP A  #$3F      IS BYTE IN TABLE RANGE?
01D4 25 E6          BCS    EAERR     IF NOT CONVERT TO SPACE
01D6 80 40          SUB A  #$40      ELSE PROCESS AS IS
01D8 20 06          BRA    LOAD      DO IT
01DA 81 FA   UPRBLK CMP A  #$FA      IS IT IN TABLE?
01DC 24 DE          BCC    EAERR     IF NOT TREAT AS SPACE
01DE 80 80          SUB A  #$80      ELSE TREAT AS IS FOR UPPER BLOCK
01E0 0F      LOAD   SEI              DISABLE INTERRUPTS
01E1 DF00           STX    EAXSAV    SAVE DATA POINTER
01E3 97 03          STA A  EANDX+1   SET UP INDEX LOWER BYTE
01E5 DE02           LDX    EANDX     LOAD UP INDEX
             *AN ERROR 15 IS EXPECTED AND ACCOUNTED FOR NEXT LINE
             *THE UPPER 8 BITS ARE TRUNCATED WHICH IS DESIRED
ER15 line 3150
01E7 A6 F5          LDA A  EATAB,X   LOAD CONVERTED BYTE FROM TABLE
01E9 DE00           LDX    EAXSAV    RELOAD DATA POINTER
01EB 0E             CLI              REENABLE INTERRUPTS
01EC A7 00   STORA  STA A  X         RETURN CONVERTED BYTE TO BUFFER
01EE 08             INX              BUMP DATA POINTER
01EF 5A             DEC B            DECREMENT BYTE COUNT
01F0 26 C0          BNE    EALOOP    LOOP UNTIL DONE
01F2 31      EARET  INS              RESTORE STACK
01F3 31             INS
01F4 39             RTS              AND RETURN
             *EXECUTION TIME APPROX 115 USEC
             *LOOP TIME APPROX 70 USEC PER CONVERSION
             *
             *
             *EATAB: EBCDIC TO ASCII TRANSLATION TABLE
             *ARRANGED IN TWO BLOCKS;$40—$7F,$C0—$F9
             *TOTAL SIZE OF TABLE: 122 BYTES
             *
             *
     01F5    EATAB  EQU    *
01F5 20             FCB    $20,$20,$20,$20,$20,$20
01F6 20
01F7 20
01F8 20
01F9 20
01FA 20
01FB 20             FCB    $20,$20,$20,$20
01FC 20
01FD 20
01FE 20
01FF 5B             FCB    $5B,$2E,$3C,$28,$2B,$5D
0200 2E
0201 3C
0202 28
0203 2B
0204 5D
0205 26             FCB    $26,$20,$20,$20,$20
0206 20
0207 20
0208 20
0209 20
020A 20             FCB    $20,$20,$20,$20,$20
020B 20
020C 20
020D 20
020E 20
020F 21             FCB    $21,$24,$2A,$29,$3B,$5E
0210 24
0211 2A
0212 29
0213 3B
0214 5E
0215 2D             FCB    $2D,$2F,$20,$20,$20
0216 2F
0217 20
0218 20
0219 20
021A 20             FCB    $20,$20,$20,$20,$20
021B 20
021C 20
021D 20
021E 20
021F 20             FCB    $20,$2C,$25,$5F,$3E,$3F
0220 2C
0221 25
0222 5F
0223 3E
0224 3F
0225 20             FCB    $20,$20,$20,$20,$20
0226 20
0227 20
0228 20
0229 20
```

the caller's registers from the stack by calling RPOP (refer to the RPOP flow diagram), and program control is passed back to the caller.

### The Interface

Actually, the subtitle is somewhat misleading, in that there are actually two interfaces. The first interface exists between the caller and MONTOR, while the second is between MONTOR and the library routine. Typically, a call to MONTOR requesting two eight-bit numbers to be multiplied by routine number twelve (multiply routine) might look like Example 2.

The bracketed instructions are the fixed format to use when calling any library routine; it is the same format no matter what information the library routine requires. What does change from routine to routine is the information that gets stored at the argument set address. In the example above, not only the two numbers to be multiplied are at the argument set address, but the final result is left there also.

In the routines I've supplied as examples, only the address of the data blocks is at the argument set address. This allows the same routine to specify different blocks of data to be operated on. Simply change the data block address which is stored at the argument set address.

At the other end of the monitor call processor is the MONTOR-library routine interface. This format is only necessary to know if you intend to write utility routines for the library, and is irrelevant information when you are just using the library routines. To see what needs to be done, study the contents of the stack carefully (Example 1), then read the discussion that follows.

The first operation necessary is to set up the return address on the stack. The correct return address consists of the top two entries on the stack, MONTOR Return

```
022A 20               FCB   $20,$20,$20,$20,$20
022B 20
022C 20
022D 20
022E 20
022F 3A               FCB   $3A,$23,$40,$20,$3D,$22
0230 23
0231 40
0232 20
0233 3D
0234 22
0235 20               FCB   $20,$41,$42,$43,$44
0236 41
0237 42
0238 43
0239 44
023A 45               FCB   $45,$46,$47,$48,$49
023B 46
023C 47
023D 48
023E 49
023F 20               FCB   $20,$20,$20,$20,$20,$20
0240 20
0241 20
0242 20
0243 20
0244 20
0245 20               FCB   $20,$4A,$4B,$4C,$4D
0246 4A
0247 4B
0248 4C
0249 4D
024A 4E               FCB   $4E,$4F,$50,$51,$52
024B 4F
024C 50
024D 51
024E 52
024F 20               FCB   $20,$20,$20,$20,$20,$20
0250 20
0251 20
0252 20
0253 20
0254 20
0255 5C               FCB   $5C,$20,$53,$54,$55
0256 20
0257 53
0258 54
0259 55
025A 56               FCB   $56,$57,$58,$59,$5A
025B 57
025C 58
025D 59
025E 5A
025F 20               FCB   $20,$20,$20,$20,$20,$20
0260 20
0261 20
0262 20
0263 20
0264 20
0265 30               FCB   $30,$31,$32,$33,$34
0266 31
0267 32
0268 33
0269 34
026A 35               FCB   $35,$36,$37,$38,$39
026B 36
026C 37
026D 38
026E 39
                *
                *
                *AETAB: ASCII TO EBCDIC TRANSLATION TABLE
                *ORGANIZED AS ONE CONSECUTIVE TABLE
                *SUBTRACTING $20 FROM THE ASCII CODE AND
                *ADDING THE RESULT TO THE TABLE BASE ADDRESS
                *WILL GIVE THE ADDRESS OF THE CORRESPONDING EBCDIC CODE
                *TOTAL TABLE SIZE: 64 BYTES
                *
                *
        026F    AETAB EQU   *
026F 40               FCB   $40,$5A,$7F,$7B
0270 5A
0271 7F
0272 7B
0273 5B               FCB   $5B,$6C,$50,$7D
0274 6C
0275 50
0276 7D
0277 4D               FCB   $4D,$5D,$5C,$4E
0278 5D
```

Adx HI (and LO). In order to maintain stack continuity, this address must be placed into the RTS Adx HI (and LO) for the library routine's return from subroutine instruction (RTS). In assembly language form the sequence to follow is shown in Example 3.

Now the routine needs to gain access to the argument set pointer and the number of argument sets. This is accomplished by the sequence in Example 4.

At this point, accumulator A gives us the count of the argument sets, and the index register points to the first argument set (if there is more than one). Executing a return from subroutine (RTS) at this point will return control to the caller via MONTOR, but normally it is at this point that the library routine would begin to process the data supplied by the caller and now pointed to by the index register.

You needn't follow this sequence exactly when writing a library routine, but if you deviate from this format use caution because mangling the stack order can produce some pretty bizarre results. If, for instance, your routine needs to keep the argument pointer for later use (specifically, multiple argument routines such as MOVBLK, a multiple memory block move routine in the works), the two INS instructions should be moved to the end of your routine so the pointer is not destroyed if an interrupt occurs during the routine's execution.

### The Utility Library

I have a multitude of ideas about a utility library, but I'm sure many of you have good ideas on this also, so I'll hold myself in check as much as possible here. Dick Wilcox had several ideas on the types of routines to put into a utility library, and with a repertoire of up to 127 routines which can call each other to build more and more elaborate functions, the utility library can become a very powerful system tool.

Some routines that I see as being generally useful have been included with the MONTOR listing, and the flowcharts have also been included for those of you interested in the guts of the routines. I've commented the listings rather verbosely for just such a case, but then I feel that the gut workings of a routine comprise knowledge that the monitor call processor makes unnecessary. You should be able to plug these routines into your system and go!

There are some other types of routines that can be very useful in program and system development, such as:

1. A system initialization routine to set up I/O devices, the system clock, software pointers, etc., when the system is reset.

2. Interrupt-driven input and output routines that allow the processor to execute programs while a peripheral is busy (no waiting on slow devices).

3. A general timekeeping routine that services the system clock and maintains a real time clock with seconds, minutes, hours, and day/date.

4. A snapshot routine that, when armed, can sample the program counter periodically on interrupt and return a histogram of program execution. (Did you ever wonder just where in a program your

```
0279 5C
027A 4E
027B 6B                FCB     $6B,$60,$4B,$61
027C 60
027D 4B
027E 61
027F F0                FCB     $F0,$F1,$F2,$F3
0280 F1
0281 F2
0282 F3
0283 F4                FCB     $F4,$F5,$F6,$F7
0284 F5
0285 F6
0286 F7
0287 F8                FCB     $F8,$F9,$7A,$5E
0288 F9
0289 7A
028A 5E
028B 4C                FCB     $4C,$7E,$6E,$6F
028C 7E
028D 6E
028E 6F
028F 7C                FCB     $7C,$C1,$C2,$C3
0290 C1
0291 C2
0292 C3
0293 C4                FCB     $C4,$C5,$C6,$C7
0294 C5
0295 C6
0296 C7
0297 C8                FCB     $C8,$C9,$D1,$D2
0298 C9
0299 D1
029A D2
029B D3                FCB     $D3,$D4,$D5,$D6
029C D4
029D D5
029E D6
029F D7                FCB     $D7,$D8,$D9,$E2
02A0 D8
02A1 D9
02A2 E2
02A3 E3                FCB     $E3,$E4,$E5,$E6
02A4 E4
02A5 E5
02A6 E6
02A7 E7                FCB     $E7,$E8,$E9,$4A
02A8 E8
02A9 E9
02AA 4A
02AB E0                FCB     $E0,$4F,$5F,$6D
02AC 4F
02AD 5F
02AE 6D
```

*JMPTAB*

```
                *JMPTAB: MONITOR SUBROUTINE LIBRARY TABLE
                *EACH SUB IS REFERENCED EXTERNALLY BY NUMBER
                *BUT LISTED BY NAME IN THE TABLE
                *JMPTAB MAY BE PLACED ANYWHERE IN MEMORY AS CONVENIENT

     02AF       JMPTAB EQU   *
02AF 0161              FDB   ASCEBC    CALL RTN 0
02B1 01A2              FDB   EBCASC    CALL RTN 1
02B3                   RMB   *—JMPTAB+256
                *INSERT NEW ROUTINES IN FRONT OF THE RMB DIRECTIVE
03B7                   END
```

***Stack Contents upon entry to Library Routine:***

- MONTOR Return Adx HI
- MONTOR Return Adx LO
- # Arguments in Argument List
- ARG LIST PTR HI
- ARG LIST PTR LO
- RTS Adx HI
- RTS Adx LO

*Example 1. Library format.*

```
 STAA   ARGLIST      (Put away multiplicand)
 STAB   ARGLIST+1    (Put away multiplier)
(JSR    MONTOR
{FCB    12           (Call Multiply Routine)
{FCB    1            (Multiply only 1 set of #s)
(FDB    ARGLIST      (Give address of arguments)
 LDX    ARGLIST      (Pick up 16 bit product in X)
```

*Example 2. MONTOR format.*

```
PUL A           (Strip MONTOR Adx HI off stack)
PUL B           (Strip MONTOR Adx LO off stack)
TSX             (X register points to stack)
STAA    3,X     (Set up 'RTS' Adx HI)
STAB    4,X     (Set up 'RTS' Adx LO)
```

*Example 3.*

```
PUL A           (ACCA now contains # of arguments)
LDX     1,X     (X REG now points to argument set)
INS             (Move SP up to 'RTS' address)
INS
```

*Example 4.*

processor spends all its time? Tighten up a few loops and it is easily possible to double your processor's throughput.)

5. Math functions.

6. Extended formatting routines to maintain files and records on the system mass storage device.

7. Conversion routines for input or output with peripherals using different character code sets.

8. A high-speed sorting routine.

9. A search routine.

10. A compare routine.

Some other utility routines I'm currently involved in developing are a high-speed sorting algorithm and a G.P.I.O. processor which will talk on the I.E.E.E. 488 standard parallel interface bus. I'd really like to see the monitor call processor used by a lot of you 6800 programmers; it seems a real waste for everybody to keep reinventing the wheel.

I find programming to be very much like chess: intense, challenging, and rewarding to do, but rather dull to study. With a library of good system routines which don't require a lot of intense study in order to implement them on a new system, programming can be that much more enjoyable . . . for us all! ■

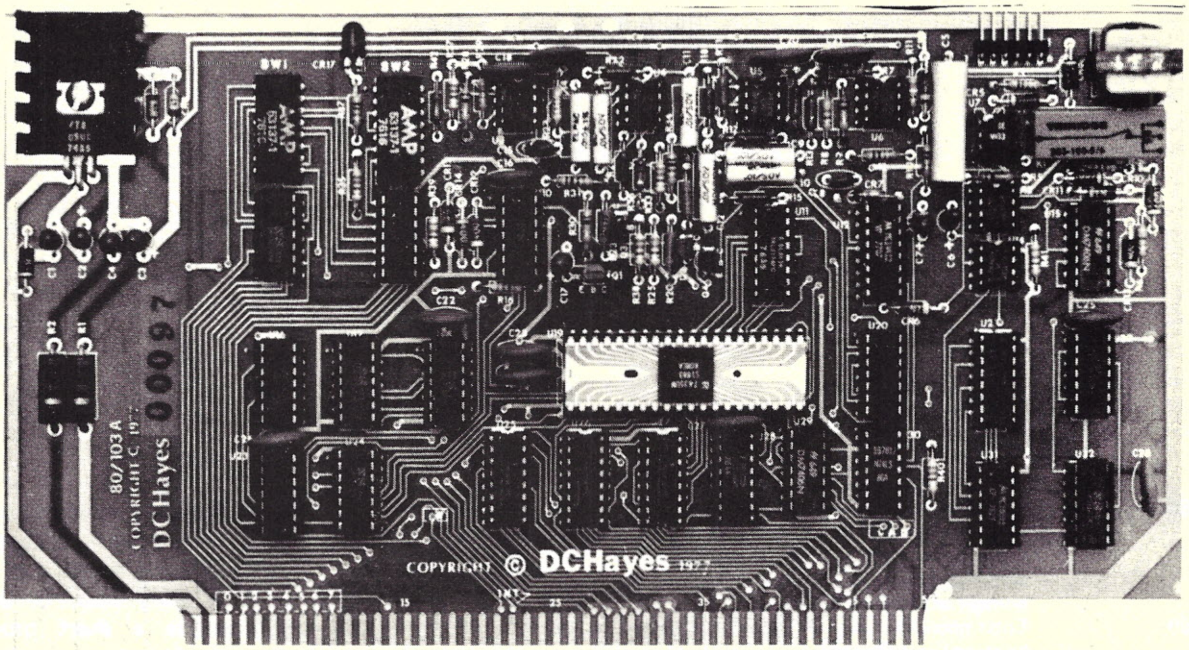
```
S00600004844521B
S113000201F5026F02AFEB01A900373630EE003181
S105001231397E
S11301008D1F8D404D2B0A48CE00068D07EE00ADA5
S1130110008D21395FAB01E900363730EE00313113
S11301203936370736OFDF009600D6010E373630E2
S1130130EE056E00323330A705E706EE00313132AA
S113014006333239 30EE09A602E6033736E60137C4
S1130150A60030C604EB0DE70D24026C0CEE036E12
S1130160000303332A706E705EE0333E602271EEE1E
S113017000A6002A02862080202B150FDF00970599
S1130180DE04A66FDE000EA700085A26E4313139DA
S11301908B20810A2604860D20ED810D26D7862525
S11301A020E5303332A706E70533EE03E6022742A3
S11301B0EE00A60081C02422818025168 10D26042C
S11301C0860A20288125260486 0D20208620201CCE
S11301D0200E813F25E68040200681FA24DE8080BF
S11301E00FDF009703DE02A6F5DE000EA700085A13
S11301F026C03131392020202020202020202020205BDF
S11302002E3C282B5D2620202020202020202020202169
S11302102 42A293B5E2D2F2020202020202020204E
S1130220 2C255F3E3F20202020202020202020203A23
S11302302340203D222041424344454647484920 2B
S11302402020202020204A4B4C4D4E4F505152200C
S1130250202020202020 5C2053545556575859 5A20AA
S11302602020202020203031323334353637383940 9D
S11302705A7F7B5B6C507D4D5D5C4E6B604B61F0D7
S1130280F1F2F3F4F5F6F7F8F97A5E4C7E6E6F7CD2
S1130290C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7A1
S11302A0D8D9E2E3E4E5E6E7E8E94AE04F5F6D0127
S10602B06101A243
S9030000FC
```

*Program B. Object code, Motorola format.*

# TIMESHARING

The 80-103A works both ways. Your system can call a timesharing service and communicate as an intelligent terminal *OR* your S-100 system can be the timesharing system where the 80-103A answers the phone and communicates with terminals or other processors.

## 80-103A DATA COMMUNICATIONS ADAPTER




*The 80-103A DATA COMMUNICATIONS ADAPTER was developed to function as an S-100 bus compatible serial interface incorporating a fully programmable modem and Telco interface. These functions are usually accomplished by the use of two separate modules: 1) a serial I/O board, and 2) an external modem. By combining these features on a single board, the 80-103A can offer microcomputer applications significant cost/performance advantages over other implementations.*

- FULLY PROGRAMMABLE FEATURES
- AUTOMATED DIALING AND ANSWER
- ORIGINATE OR ANSWER MODE
- 110-300 BIT/SEC DATA RATES
- CHARACTER FORMAT AND PARITY
- ERROR DETECTION
- FULLY BUFFERED, OUTPUTS DRIVE 25 S-100 BUS LOADS
- STANDARD U.S. FREQUENCIES
- FULL TELCO COMPATIBILITY WHEN ATTACHED TO DAA
- COMPATIBLE WITH EXISTING TELETYPES AND TIME SHARING MODEMS
- ALL DIGITAL MODULATION AND DEMODULATION. NO ADJUSTMENTS REQUIRED.

DC Hayes Associates offers a full range of capabilities for solving your information handling problems. Whether your problem is large or small, we will apply innovative techniques for finding the best solution. Contact us about our products and services.

PRICES:
**Assembled 80-103A with 48 hour burn in and 90 day warranty is $279.95**

**Bare Board with manual is $49.95**

BANKAMERICARD VISA ACCEPTED

H20

# D C Hayes associates

**P.O. Box 9884 • Atlanta, Georgia 30319 • (404) 231-0574**

# Memory Debugging

## which chip is it?

| | |
|---|---|
| 0C80 - 0C85 | Set lowest and highest addresses to be checked. It is best to confine testing to one board at a time as program execution time increases exponentially with the number of locations checked. |
| 0C86 - 0C89 | Enter maximum and minimum numbers to be stored and checked. |
| 0C8A | Reset the carry flag as we will need it later. |
| 0C8B - 0C8C | Move the number to be stored to the memory location indicated by HL, and then see if it was actually held there. |
| 0C8D - 0C8F | If it was not accepted, stop the program. The row location of the bad chip can be found in HL. |
| 0C90 | Move the 1 bit one place to the left in preparation for testing the next bit in memory. |
| 0C91 - 0C93 | If we haven't moved the bit into the carry flag, go back and store it in memory. |
| 0C94 - 0C9D | Find out if we have checked all the locations we wanted to. If not, jump to the cross-feed checkout routine. |
| 0C9E - 0CA1 | If we have checked all locations, HALT and make sure that an interrupt has not interfered with the HALT. |
| 0CA2 - 0CA4 | Store the last memory location in a place we are not using or testing. |
| 0CA5 - 0CA6 | Check to see if 80 is still in the last location. |
| 0CA7 - 0CA9 | If it isn't, HALT. The location held in HL is the receiver of the cross feed, and the location held in 0CC1 is the sender. |
| 0CAA - 0CB5 | Check to see if we have worked our way back to the beginning of the RAM board. If we haven't, drop down one more location and check again for cross feed. |
| 0CB6 - 0CBC | If we have gone all the way to the beginning, return our original HL and go back to the memory-acceptance part of the program. |

*Table 1. Program explanation.*

*Albert Brunelli*
*RFD #1*
*Berlin NH 03570*

The idea of checking RAM boards first occurred to me after I had assembled my Polymorphic Systems Poly 88 System 6 and was running some of the sample programs in BASIC. One program would not run correctly because a wrong symbol was entered on one line. I tried changing the line to the correct symbol, but the same error recurred. By changing the symbol entered at the location of the error and noting the result that appeared after a LIST instruction, I was able to determine that a number was permanently stored in one of the RAM locations and was being added to whatever I entered.

The next problem was to find the location of the bad memory chip. To do this, I wrote a short program in machine language that would put 00 hex in each location and then check to see if it could be brought back. The program worked and told me that the bad location was 4C97, and that the number permanently stored was 02. The problem now was to find the chip that held that location and that number.

The literature that came with the RAM board was some help. It informed me that the chips were 1K x 1 and that the addresses and bits were arranged by row and column. Since it did not specify top, bottom, right or left, I had to move chips around in suspected areas until the location of the error moved. This was a tedious process. It need not be repeated, as I have included the chip arrangement in Fig. 1.

After a call to Microcomputers Inc. in Nashua NH, I discovered that there were

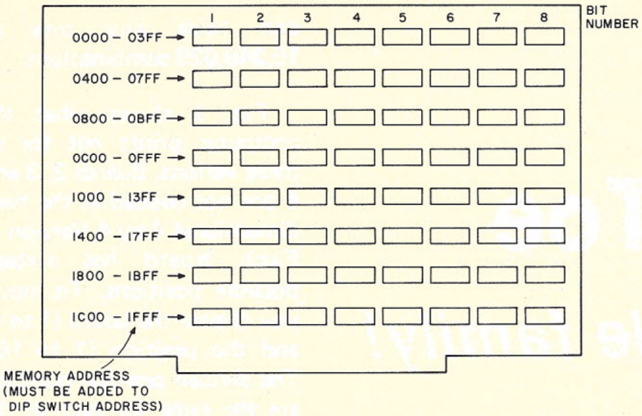



*Fig. 1.*

| Step no. | Code (hex) | Mnemonic | Comment |
|---|---|---|---|
| 0C80 | 21 | LXI H | Load lowest address |
| 0C81 | | | to be checked. |
| 0C82 | | | |
| 0C83 | 11 | LXI D | Load highest address |
| 0C84 | | | to be checked. |
| 0C85 | | | |
| 0C86 | 0E | MVI C | Highest number to |
| 0C87 | 80 | | be stored. |
| 0C88 | 3E | MVI A | Lowest number to |
| 0C89 | 01 | | be stored. |
| 0C8A | B7 | ORA A | Reset carry. |
| 0C8B | 77 | MOV M,A | Store number. |
| 0C8C | BE | CMP M | See if accepted. |
| 0C8D | C2 | JNZ | Halt if not accepted. |
| 0C8E | 9E | | |
| 0C8F | 0C | | |
| 0C90 | 17 | RAL | Move bit left. |
| 0C91 | D2 | JNC | |
| 0C92 | 8B | | |
| 0C93 | 0C | | |
| 0C94 | 7D | MOV A,L | See if we have |
| 0C95 | BB | CMP E | reached last |
| 0C96 | C2 | JNZ | location specified. |
| 0C97 | A2 | | |
| 0C98 | 0C | | |
| 0C99 | 7C | MOV A,H | Check high address to see if done. |
| 0C9A | BA | CMP D | |
| 0C9B | C2 | JNZ | |
| 0C9C | A2 | | |
| 0C9D | 0C | | |
| 0C9E | 76 | HLT | Halt. |
| 0C9F | C3 | JMP | |
| 0CA0 | 9E | | |
| 0CA1 | 0C | | |
| 0CA2 | 22 | SHLD | Temporarily store |
| 0CA3 | C1 | | present location. |
| 0CA4 | 0C | | |
| 0CA5 | 79 | MOV A,C | See if 80 is |
| 0CA6 | BE | CMP M | still in memory. |
| 0CA7 | C2 | JNZ | Halt if it isn't. |
| 0CA8 | 9E | | |
| 0CA9 | 0C | | |
| 0CAA | 7D | MOV A,L | See if we are at the |
| 0CAB | FE | CPI | beginning of the RAM. |
| 0CAC | 00 | | Low byte of |
| 0CAD | C2 | JNZ | lowest address |
| 0CAE | BD | | |
| 0CAF | 0C | | |
| 0CB0 | 7C | MOV A,H | |
| 0CB1 | FE | CPI | |
| 0CB2 | | | High byte of lowest |
| 0CB3 | C2 | JNZ | address. |
| 0CB4 | BD | | |
| 0CB5 | 0C | | |
| 0CB6 | 2A | LHLD | Get back location |
| 0CB7 | C1 | | for acceptance part. |
| 0CB8 | 0C | | |
| 0CB9 | 23 | INX H | |
| 0CBA | C3 | JMP | |
| 0CBB | 88 | | |
| 0CBC | 0C | | |
| 0CBD | 2B | DCX H | |
| 0CBE | C3 | JMP | |
| 0CBF | A5 | | |
| 0CC0 | 0C | | |
| 0CC1 | | | Temporary storage |
| 0CC2 | | | for memory location. |

*Program listing.*

other possible memory problems. One of the more difficult is cross feed between memory locations. In this situation, data loaded at one location will alter the contents of another location. The program below will determine if all locations will hold data and if there is any cross feed to other locations.

### Program Operation

The program works as follows: 00000001 is sent to the first memory location and then recalled to be sure that it was accepted by the RAM. If it was, the accumulator is rotated left to give 00000010, for which the storage and retrieval are repeated.

When we get to the point where we have rotated the 1 out of the accumulator and into the carry flag, we test all previously loaded locations to be sure that the contents have not changed.

If a location refuses to accept data, the program will halt. The contents of the H register will tell you the row in which the bad chip is located. The number in the memory location should tell you which column the chip is in. If it doesn't, try storing 00 in the location given by HL. If it accepts 00, try FF. The number that appears will indicate which column contains the bad chip. For example, when attempting to store 00, if you find 01, then a 1 was stored and the bad chip is in the first column on the left. If you find 40, then a 1 is stuck in the second column from the right.

Now, if a cross feed has occurred, the program will stop at the receiver of the cross feed, and its location will be held in HL. The location of the sender of the cross feed will be held at 0CC1. The bit that was cross fed should indicate the column in which the problem lies. See Table 1 for an explanation of the program.

There is one chance in sixteen that the program will miss a cross-feed problem. This chance is that a 1 has been fed into the most significant bit (MSB) of a lower location. If you have a memory problem that the program does not find, I suggest you modify the steps in Example 1 and run it again. ■

| | | |
|---|---|---|
| 0C87 | 01 | |
| 0C89 | 80 | |
| 0C90 | 1F | RAR |

*Example 1.*

*Joseph Roehrig*
*Box 74*
*Middle Village NY 11379*

# 3-D Tic-Tac-Toe

## *a winner with the whole family!*

**N**ow that you have your computer running, it is time to entertain your family and friends. At the same time, you should impress them with your computer's brilliance. A game is the natural medium to introduce others to your new sophisticated toy and a familiar game is a wise choice. Tic-tac-toe is very well-known and a logical choice for your demonstration.

IBM had a tic-tac-toe game in its pavilion at the 1964 New York World's Fair. IBM's game could never lose, but also could never win against a knowledgeable player due to the simplicity of the game. A standard tic-tac-toe game has a two-dimensional 3-box by 3-box game board. There are only nine possible moves, making the game rather easy to play for both man and machine.

To improve our display game, the board has been expanded from the standard 3 by 3 to a 4 by 4. This adds to the complexity of the game, but a skilled player can still stand off a computer. Here we add a third dimension and increase the size of the board. Now the game becomes a real challenge with its 4 by 4 by 4 cubic look.

### Three Versions

This article describes and provides programming details for three versions of tic-tac-toe. All three versions will be derived from one relatively short sixty line BASIC language source listing. The program is written in Altair 3.2 BASIC and is geared for a video terminal with 80 characters and 24 lines. (An option to the coding is described to reduce the printed output if desired.)

The versions are:

Version 1 – a simple 4 by 4 with quick response time.

Version 2 – a 4 by 4 by 4 which can be beaten and respond to each move in 25 seconds or less.

Version 3 – an almost unbeatable 4 by 4 by 4 game. I'm forced to say almost because the possible moves can go as high as 64 factorial. An easier way of pointing out the numerous possibilities is to say that the first two moves taken by each player can take any one of 15,249,025 combinations.

Fig. 1 shows what the computer prints out for all three verions. Boards 2, 3 and 4 are not needed in the two-dimensional 4 by 4 Version 1. Each board has sixteen possible positions. To move, you choose the board (1 to 4) and the position (1 to 16). The sixteen position numbers are the same for each of the four boards.

### Version 3 – The Rough One!

An easy way to proceed is to describe Version 3, the most complex, and then discuss the modifications necessary to use the other two. Fig. 2. is the flowchart of the game. First, the computer sets all of its 64 board squares to double blanks. During the game, the computer's squares will be marked CC and the player's UU. The board is different from Fig. 1. The computer really has only one board that contains 64 squares. Fig. 3 shows how the computer's 64 squares correspond to the player's four boards of sixteen squares each. Also detailed in Fig. 3 are the 76

```
YOUR MOVES ARE UU AND I'M CC
POSITION # ARE
                                        1   5   9  13
                                        2   6  10  14
                                        3   7  11  15
                                        4   8  12  16

   BOARD 1       BOARD 2       BOARD 3       BOARD 4

   UU/  /  /       /  /  /       /  /  /       /  /  /
   ///////////   ///////////   ///////////   ///////////
     /  /  /     CC/  /  /       /  /  /       /  /  /
   ///////////   ///////////   ///////////   ///////////
     /  /  /       /  /  /       /  /  /       /  /  /
   ///////////   ///////////   ///////////   ///////////
     /  /  /       /  /  /       /  /  /       /  /  /

      YOUR BOARD, POSITION? 1,4
      I WANT BOARD 1 POSITION 2

      (EVERYTHING EXCEPT LINE 1 IS REPRINTED UP TO
      ''YOUR BOARD ...'')
```

*Fig. 1. Display of game being run.*

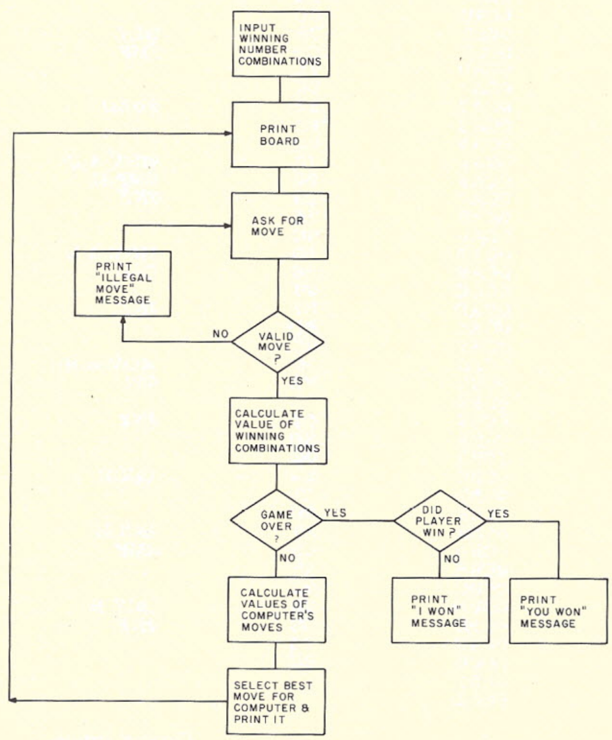



*Fig. 2. Tic-tac-toe flowchart.*

possible winning combinations that must be stored in the computer's memory. Only winning moves 1 to 10 are needed for the two-dimensional game. Wins 1 to 40 have moves that are all on one board. The winning moves described as 41 to 76 have one move on each of the four boards. Some of these are tricky and a study of Fig. 3 will familiarize you with all of the possible ways of winning.

The computer then asks for your move (Fig. 1). Your move is a board number, a comma, and a position number. These two figures are converted into a computer position which is described in (Fig. 3). If you move to an occupied space or type in an invalid move (not a number from 1 to 4 followed by a comma and a number from 1 to 16), the computer will again ask for your move. Note: in the two-dimensional game, a move to any board other than board 1 will result in a lost move.

The computer now calculates the value of each of the 76 possible winning combinations. The value is equal to the sum of the values assigned to each of the four squares or board positions contained in the winning combination. The values of the board positions are:

0 for an unoccupied box — prints 2 blanks on game board.

1 for your boxes — prints UU on appropriate game board position.

5 for computer occupied boxes — prints CC on game board.

These values are important and are used in all move decisions.

The computer now sees if the value 4 exists in any of the 76 win possibilities. If 4 exists, you have beaten the computer and the game is over. A four designates a player's win, since the only way four can exist is to have a 1, a player's box, in each of

```
10 DIM S(64), W(3,76), S$(64), V(76)
20 FOR A = 1 TO 10 : FOR A1 = 0 TO 3 : READ W(A1,A) : NEXT A1,A
30 FOR A = 1 TO 3 : A1*10 : FOR A2= 1 TO 10 : FOR A3 = 0 TO 3
40 W(A3,A1+A2)=W(A3,A2)+(16*A) : NEXT A3,A2,A
50 FOR A = 41 TO 56: FOR A1 = 0 TO 3
60 W(A1,A) = (A1*16)+A-40 : NEXT A1,A
70 FOR A = 57 TO 76 : FOR A1 = 0 TO 3 : READ W(A1,A) : NEXT A1,A
72 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,1,5,9,13,2,6,10,14
74 DATA 3,7,11,15,4,8,12,16,1,6,11,16,4,7,10,13,1,22,43,64
76 DATA 5,22,39,56,9,26,43,60,13,26,39,52,2,22,42,62,14,26,38,50
78 DATA 3,23,43,63,15,27,39,51,4,23,42,61,8,23,38,53,12,27,42,57,16,27,38,49
80 FOR A = 1 TO 64: S$(A)=" " : S(A)=0 : NEXT A
83 DATA 1,21,41,61,1,18,35,52,4,19,34,49,4,24,44,64
84 DATA 13,25,37,49,13,30,47,64,16,31,46,61,16,28,40,52
85 PRINT"YOUR MOVES ARE UU AND I'M CC"
90 GOSUB 1000 : "INPUT "YOUR BOARD, POSTION";A1,A2
100 A=((A1-1)*16)+A2
105 IF A >64 OR >1 A THEN PRINT"ILLEGAL MOVE" : GOTO 90
110 IF S(A) <>0 THEN PRINT"YOU CAN'T MOVE THERE" : GOTO 90
120 S(A)=1 : S$(A)="UU"
190 M5=0 : FOR A = 1 TO 76
192 A2= W(0,A) : A3=W(1,A) : A4= W(2,A) : A5=W(3,A)
194 V(A)=S(A2)+S(A3)+S(A4)+S(A5)
196 IF V(A)=4 THEN 410
198 IF V(A) = 15 THEN M5=A
199 NEXT A : IF M5 <>0 THEN 365
200 M3=9
204 Y1=0
205 FOR A = 1 TO 64 : M2=0
210 IF S(A) <>0 THEN 350
215 Y1 = Y1+1
220 FOR A1 = 1 TO 76
225 FOR A2 = 0 TO 3 : IF A=W(A2,A1) THEN 230
228 NEXT A2: GOTO 300
230 A6= V(A1)
260 IF A6=3 THEN M4=A : GOTO 390
270 IF A6 = 0 THEN 300
280 IF 5 >A6 THEN M2 = M2 + A6 A6 : GOTO 300
290 A7= INT(A6/5) : IF A7 = A6/5 THE M2=M2+A7
300 NEXT A1
320 IF M2 >M3 THEN M3=M† : M4=A
350 NEXT A : GOTO 390
365 FOR A1 = 0 TO 3 : A6 = W(A1,M5) : IF S(A6) = 0 THEN M5=A6 : GOTO 368
367 NEXT A1
368 PRINT "THE OLD"; M5
370 S$(M5)="CC" : A1= INT(M5-1/16)+1 : A2= M5-((A1-1)*16)
380 PRINT"I WON WITH BOARD";A1;"POSITION";A2 : GOSUB 1000
382 INPUT"READY";A1 : GOTO 80
390 S$(M4)="CC" : S(M4)=5
392 A1=INT((M4-1)/16) : A2=M4-((A1-1)*16)
400 PRINT"I WANT BOARD";A1;"POSITION";A2 : GOTO 90
410 PRINT : PRINT"YOU WON" : GOSUB 1000 : GOTO 80
1000 PRINT"POSITIONS ARE"; : FOR A + 0 TO 3 : FOR A1=1 TO 13 STEP 4 : A2=20 + (A1*4)
1100 PRINT TAB(A2); A+A1 ; : NEXT A1 : PRINT : NEXT A :PRINT:PRINT
1105 FOR A=0 TO 3 : PRINT TAB(A*15);"BOARD";A+1; : NEXT : PRINT
1107 PRINT : PRINT
1110 FOR A = 1 TO 4 : A1 = 0 TO 48 STEP 16 : A2 = A +A1
1120 PRINT S$(A2);"/";S$(A2+4);"/";S(A2+8);"/";S(A2+12);" ";: NEXT A1
1125 IF A=4 THEN 1130
1127 PRINT : FOR A2 = 1 TO 4 : PRINT"////////// "; : NEXT A2
1130 PRINT : NEXT A : PRINT : PRINT : RETURN
```

*Program A. BASIC program for Verson 3 of three-dimensional tic-tac-toe.*

```
190 M5=0 : Q=0 : FOR A = 1 TO 76
197 IF V(A)=3 THEN Q=A
201 IF Q=0 THEN 205
202 FOR A9= 0 TO 3 : A6 =W(A9,Q) : IF S(A6)=0 THEN M4=A6 : GOTO 390
203 NEXT A9
220 FOR A1= 60 TO 76 STEP 2
280 IF 5 >A6 THEN M2=M2+A6 : GOTO 300
350 NEXT A : IF M3 <>0 THEN 390
352 FOR A =1 TO 64 : IF S(A)=0 THEN M4=A : GOTO 390
354 NEXT A
```

Note: In line 280, change M2=M2+A6 to M2=M2+A6↑A6 to make this version harder to beat.

*Program B. Modifications necessary to obtain Version 2 of the game.*

**Computer Boards:**

| Board #1 | Board #2 | Board #3 | Board #4 |
|---|---|---|---|
| 1 5 9 13 | 17 21 25 29 | 33 37 41 45 | 49 53 57 61 |
| 2 6 10 14 | 18 22 26 30 | 34 38 42 46 | 50 54 58 62 |
| 3 7 11 15 | 19 23 27 31 | 35 39 43 47 | 51 55 59 63 |
| 4 8 12 16 | 20 24 28 32 | 36 40 44 48 | 52 56 60 64 |

**Winning moves per computer boards:**

| | | | |
|---|---|---|---|
| 1) | 1 2 3 4 | 39) | 49 54 59 64 |
| 2) | 5 6 7 8 | 40) | 52 55 58 61 |
| 3) | 9 10 11 12 | 41) | 1 17 33 49 |
| 4) | 13 14 15 16 | 42) | 2 18 34 50 |
| 5) | 1 5 9 13 | 43) | 3 19 35 51 |
| 6) | 2 6 10 14 | 44) | 4 20 36 52 |
| 7) | 3 7 11 15 | 45) | 5 21 37 53 |
| 8) | 4 8 12 16 | 46) | 6 22 38 54 |
| 9) | 1 6 11 16 | 47) | 7 23 39 55 |
| 10) | 4 7 10 13 | 48) | 8 24 40 56 |
| 11) | 17 18 19 20 | 49) | 9 25 41 57 |
| 12) | 21 22 23 24 | 50) | 10 26 42 58 |
| 13) | 25 26 27 28 | 51) | 11 27 43 59 |
| 14) | 29 30 31 32 | 52) | 12 28 44 60 |
| 15) | 17 21 25 29 | 53) | 14 30 46 62 |
| 16) | 18 22 26 30 | 54) | 13 29 45 61 |
| 17) | 19 23 27 31 | 55) | 15 31 47 63 |
| 18) | 20 24 28 32 | 56) | 16 32 48 64 |
| 19) | 17 22 27 32 | 57) | 1 22 43 64 |
| 20) | 20 23 26 29 | 58) | 5 22 39 56 |
| 21) | 33 34 35 36 | 59) | 9 26 43 60 |
| 22) | 37 38 39 40 | 60) | 13 26 39 52 |
| 23) | 41 42 43 44 | 61) | 2 22 42 62 |
| 24) | 45 46 47 48 | 62) | 14 26 38 50 |
| 25) | 33 37 41 45 | 63) | 3 23 43 63 |
| 26) | 34 38 42 46 | 64) | 15 27 39 51 |
| 27) | 35 39 43 47 | 65) | 4 23 42 61 |
| 28) | 36 40 44 48 | 66) | 8 23 38 53 |
| 29) | 33 38 43 48 | 67) | 12 27 42 57 |
| 30) | 36 39 42 45 | 68) | 16 27 38 49 |
| 31) | 49 50 51 52 | 69) | 1 21 41 61 |
| 32) | 53 54 55 56 | 70) | 1 18 35 52 |
| 33) | 57 58 59 60 | 71) | 4 19 34 49 |
| 34) | 61 62 63 64 | 72) | 4 24 44 64 |
| 35) | 49 53 57 61 | 73) | 13 25 37 49 |
| 36) | 50 54 58 62 | 74) | 13 30 47 64 |
| 37) | 51 55 59 63 | 75) | 16 31 46 61 |
| 38) | 52 56 60 64 | 76) | 16 28 40 52 |

*Fig. 3. Computer's board and winning moves.*

To beat the computer, the player should establish a situation where three of his boxes are not strung together until there are at least two opportunities established by the string of three. The player establishes the UUs and then the XX making it impossible for the computer to block both winning combinations.

```
 /   /   /UU
////////////
 /   /   /
////////////
UU/XX/UU/
////////////
UU/  /   /
```

*Fig. 4. Winning strategy.*

| Value of Winning Combination | Points Awarded to Box |
|---|---|
| 1 | 1 |
| 2 | 4 |
| 3 | no points, moves to block |
| 5 | 1 |
| 10 | 2 |
| all others | 0 |

*Table 1.*

the boxes making up a winning combination.

If this condition does not exist, the program continues. The computer now checks to see if a 15 occurred during the previous evaluation. If it exists, the computer wins on this move. Fifteen — not twenty — is a winner, since unlike the player, the computer has not selected its move. Thus 15 means the computer has three boxes in a winning combination that has one unoccupied square. All the computer must do to win is find the box in the combination that equals zero and then designate that box as its move. Naturally, sixteen represents a block by the player. If the computer does not win, we continue and the computer selects its move.

The evaluation of the computer's move considers each of the sixty-four possible boxes that are unoccupied. The computer checks every unoccupied box to see which of the 76 winning moves contains that box. For each winning move containing the box or square under consideration, points are given to the box's evaluation (see Table 1).

Only five values of the winning combinations are of any importance in the evaluation process and a 3 causes an automatic move to block. The block is always taken, since it has already been established that the computer cannot win on this particular move and an unblocked 3 gives the player the opportunity of winning on the next move. Therefore, the 3 must be blocked. The values labeled "all others" are 6, 7, 11, etc. They have no value, since both a UU and a CC are already found in the particular winning combination. Thus, neither the player nor the computer can win with the combination in question.

A winning combination having two of the player's moves in it and two unoccupied squares is awarded four points. This is more than a combination having two of the computer's moves in it and two unoccupied boxes, which is awarded only two points. This is done because the player always moves ahead of the computer, forcing the computer to play defense more than offense. This priority also avoids the situation shown in Fig. 4 — the strategy that can beat the computer if this priority is not employed.

All of the possible points are totaled for every winning combination involving the box under evaluation. The box receiving the highest point score becomes the computer's move.

That completes a description of Version 3 of the tic-tac-toe game which I feel cannot be beaten by a human opponent.

Now for the bad news; it

**Version 2 Game**

| Player's Move | Computer's Move |
|---|---|
| 1,4 | 2,8 |
| 1,13 | 2,10 |
| 1,11 | 1,14 |
| 1,3 | 3,6 |
| 1,7 | 4,2 |
| Computer Wins | |

**Version 2 Game**

| Player's Move | Computer's Move |
|---|---|
| 1,4 | 2,8 |
| 1,13 | 2,10 |
| 1,11 | 1,14 |
| 1,3 | 3,6 |
| 4,2 | 1,8 |
| 1,7 | 1,10 |
| 1,15 | |
| Player Wins | |

**Version 3 Game**

| Player's Move | Computer's Move |
|---|---|
| 1,4 | 1,1 |
| 1,13 | 1,7 |
| 2,4 | 4,4 |
| 2,13 | 4,13 |
| 2,11 | 1,16 |
| 2,3 | 2,7 |
| 3,4 | 4,1 |
| 3,13 | 2,1 |
| 3,1 | 3,3 |
| 3,14 | 2,2 |
| Computer Wins | |

*Fig. 5. Sample games.*

takes about three minutes for the computer to calculate its first move. The time required to select the computer's move gradually goes down as the game progresses.

### Version 2 – Can Be Beaten

Version 2 is also three-dimensional, however, it moves in 25 seconds or less and you can beat it. The game is played exactly like Version 3, except: a) All of the combinations are not evaluated. Therefore, the computer does not always pick its best move and it is much faster. b) The computer plays offense equal to defense and does not always prevent the player's winning strategy (Fig. 4).

This version is the one I recommend that you use. Fig. 5 shows the moves from a few sample games. In the first example the player loses to Version 2; next the player beats Version 2; and last the winning moves are pitted against Version 3 without any success for the player. Depending on the skill of the player, Version 2 can be difficult to beat. In fact, it can only be beaten by the strategy described in Fig. 4. After a friend has lost a game or two to Version 2, you can then take over and easily conquer the computer much to your friend's surprise.

On all computer wins, the computer prints the old XX. XX represents the winning combination as detailed in Fig. 3.

The first version of the game is two-dimensional and needs no explanation; it is played like the other two. However, as mentioned earlier, do not move to boards 2, 3, or 4, or your turn will be lost.

### The Program Listings – And Modifications

Program A is the source listing for Version 3. Program B lists the additions, deletions and changes necessary for Version 2. Only winning combinations 60, 62, 64, 66, 68, 70, 72 are checked in Version 2. By changing line number 220 you can change the combinations (and the number of combinations) used during the execution of the program. I suggest always using combinations above 40 in order to give the computer a better chance of winning. These combinations all use the third dimension, therefore making the human player's defense more difficult.

To play the two-dimensional game, Version 1, changes shown in Program C must be incorporated into Program A.

As mentioned earlier, all versions of the game are designed for use on a video terminal and the games display the boards after every two moves (one by the player and one by the computer). They also print the computer's moves. Therefore, to avoid the board printings only, make the following change: 1000 RETURN.

You will now have to keep track of the boards on your own, unless you have a memory equal to that of the computer.

If you enter the program into your computer, you will probably make typographical errors. The two lines in Program D will help you determine whether the proper winning combinations are entered. These lines will be the first inputs you encounter.

In order to rejoin a game at any given point during debugging or for other reasons, use the lines in Program E.

These lines allow you to enter all player and computer positions. First, enter all player positions into the computer using the computer's format (numbers 1 to 64) and a 0 to stop. The same procedure is then repeated for the computer.

The program was run with 12192 words of memory. Including the BASIC, about 2500 words are left for your enhancements.

Good luck, and if you beat my almost unbeatable game, change line 290 as shown in Example 4. This will make the computer play even more defensively. I have not yet beaten Version 3, so I have not tried this modification. Figs. 6 and 7 will be an aid in debugging and enhancing the program, since they describe the major variables and computer functions by line number. ■

S(64) – value of all boxes.
S$(64) – board character for each box.
W(3,76) – 76 winning combinations.
V(76) – value of winning combinations.
M5 – if positive, possible computer win if loss has not occurred.
M2 – accumulates value of all combinations that involve the box being evaluated.
M3 – highest value for a box thus far.
M4 – number of the box having the above highest value.
A6 – value of combination being examined.

*Fig. 6. Major variables.*

10 – dimension variables.
20 to 84 – read winning combinations as data or calculate them from existing winning combinations.
80 to 120 – player's move.
190 to 199 – evaluate winning combinations and check for computer loss or possible win.
200 to 350 – select computer's move.
365 to 410 – print move information.
1000 to 1130 – subroutine to print board.

*Fig. 7. Functions by line number.*

```
190 M5=0 : FOR A = 1 TO 10
205 FOR A = 1 TO 16 : M2=0
220 FOR A1 = 1 TO 10
```

*Program C. Changes to obtain Version 1.*

```
81 INPUT A,B : FOR C= A TO B : FOR D = 0 TO 3 : PRINT W(D,C);: NEXT : PRINT: NEXT C
82 INPUT Q : IF Q>1 THEN 81
```

*Program D. Entry program for Winning Combinations data.*

```
81 INPUT Q : IF Q<>0 THEN S(Q)=1 : S$(Q)="UU" : GOTO 81
82 INPUT Q : IF Q = 0 THEN S(Q)=5 : S$(Q)="CC" : GOTO 82
```

*Program E. Modifications necessary to have "interrupted" game.*

```
290 A7=INT(A6/5) : IF A7+ A6/5 THEN M2=M2+A7-1
```

*Program F. Statement to obtain the ultimate unbeatable game.*

*Allen S. Krieger*
*44 Webster Rd.*
*Lexington MA 02173*

# Programmed Instruction Made Easy: Tiny PILOT

## *Part 2: developing your own version*

In the first part of this series, I described a home computer version of PILOT, a dialogue-oriented programmed instruction language, specifically designed for nonmathematical programming. It is, possibly, the easiest computer language to learn and use today. In this article, by describing the interpreter that I wrote, called KTP, I will show you how to write a PILOT interpreter for your own computer. The routines in the KTP interpreter are illustrated through flowcharts. After all, a Z-80 assembly-language listing wouldn't be very useful for someone with a 6800 or 6502 CPU chip, and flowcharts are the universal language of computing.

### Review

For reference purposes, Table I (reproduced from the first article) shows the 14 Tiny PILOT instructions in the KTP interpreter. They are slightly different from the instructions of the parent language. In part, these differences make programming in Tiny PILOT easier for novice programmers and also make it easier to write the interpreter.

A Tiny PILOT program consists of *statements*, each having a label (optional), a one- or two-letter *instruction* (mandatory), a colon (mandatory), an *operand field* (mandatory for some instructions, optional for others) and a carriage return (mandatory). The operand field is the text between the colon and the carriage return. It may contain text to be displayed, a variable name, a statement label name, one or more matchstrings, a counter name or the test of a mathematical relationship. The KTP interpreter acts on the operand field according to the particular instruction.

| NAME | SYMBOL | FORMAT | |
|---|---|---|---|
| type | T | (%LABEL/) | T:text (/VRBLE/) (text)*CR* |
| ask | A | " | A:(/VRBLE/) *CR* |
| match | M | " | M:/MATCHSTRING/ (,/MATCHSTRING/)*CR* |
| yes | Y | " | –Y:x . . . x*CR* |
| no | N | " | –N:x . . . x*CR* |
| jump | J | " | J:/LABEL/*CR* |
| use | U | " | U:/LABEL/*CR* |
| return | R | " | R:*CR* |
| end | E | " | E:*CR* |
| zero | Z | " | Z:n*CR* |
| bump | B | " | B:n*CR* |
| examine | X | " | X:n = or < or > ccc*CR* |
| clear | C | " | C:*CR* |
| ignore | I | " | I:text*CR* |

**DEFINITIONS**

| | |
|---|---|
| ( . . . ) | Anything within parentheses is optional. |
| %LABEL/ | A statement label name of 1 to 5 characters preceded by %, and followed by /. |
| /VRBLE/ | A variable name of 1 to 5 characters preceded and followed by slashes. |
| text | Any ASCII character string that does not include a colon or a slash. |
| /MATCHSTRING/ | An ASCII character string of one to 15 characters preceded and followed by slashes. |
| -:x . . . x*CR* | Any Tiny PILOT statement (for use with Y or N). |
| n | I, J, K or L (in Counting Instruction statements). |
| ccc | Any positive, decimal integer constant between 1 and 255 (in the X statement). |

*Table 1. Tiny PILOT Instructions.*

### The Structure of the Interpreter

To the novice programmer, an interpreter is a little gremlin that sits inside the box with the blinking lights attached to the input/output terminal. The gremlin reads the program line by line and types, asks for input, matches answers, etc., as required. The gremlin must be able to distinguish the instructions from the text in the operand field and must know what to do in response to the particular instruction it finds. Often the gremlin will have to perform similar tasks in response to several different instructions. In order to perform some tasks, such as typing a character on the output device, the interpreter should be able to call on utility routines provided as part of the operating system with your computer.

The interpreter, therefore, consists of a command processor, instruction-processing routines, interpreter utility routines and calls to system utility routines. The command processor finds the instructions, decodes them and transfers control to the appropriate instruction processor routine, which performs specific tasks and then returns control to the command processor. Both the command processor and the instruction processor can call interpreter utility routines that find the next occurrence of a specific character, load a buffer storage area or compare two character strings. Some of the instruction processors call system utility routines to carry out the detailed tasks of input and output. Fig. 1 shows the organization of the KTP interpreter.

You will be able to use the same logical flow in your command processor and instruction routines, regardless of the type of CPU. You will have to write your own utility routines, which tend to be relatively machine dependent. Owners of 8080/Z-80 or 6800 CPUs will find good examples of most KTP utility routines in the *Scelbi Software Gourmet Guide & Cookbook* series (reference 3).

### The Command Processor

In the KTP interpreter the command processor is part of the main program. Fig. 2 is a flowchart of the main program. First, the main program calls the initialization routine. This subroutine, named CLRVAR, clears the variable storage area and the variable name table. It initializes the number of active variables to zero and resets the matchflag and the active subroutine flag to "no." Finally, it initializes the primary text-pointer storage word, named MARKER, to the address of the first character of the text. The initialization routine then returns.

The actual command processor begins by finding the next (or the first) instruction. Actually, it doesn't find the instruction itself but finds the colon that follows the instruction. This is the only way that the command processor can distinguish an instruction from an identical character in the operand field of a statement.

The colon is found by a utility routine named SCANNR, the most widely used subroutine in the KTP interpreter. SCANNR requires the specification of a text pointer and two characters, a *target* and a *terminator.* The pointer is a register or two-byte memory word that holds the address of a character in the text. A flowchart of SCANNR is shown in Fig. 2b. It returns the address of the next occurrence of the target as the value of the text pointer. If SCANNR encounters the terminator before finding the target, it raises an error condition flag (e.g., the carry flag) and returns. The pointer value is incremented at the start of the routine because when the command processor calls SCANNR, the text pointer usually contains the address of the previous colon.

The command processor sets the colon as the target for SCANNR and the # as the terminator. Both KTP and my tiny text editor use # as the end-of-text indicator. If SCANNR returns with the terminator flag set, then there is no more Tiny PILOT text. The command processor acts according to an E instruction. If SCANNR has found a colon, its address is saved in MARKER.

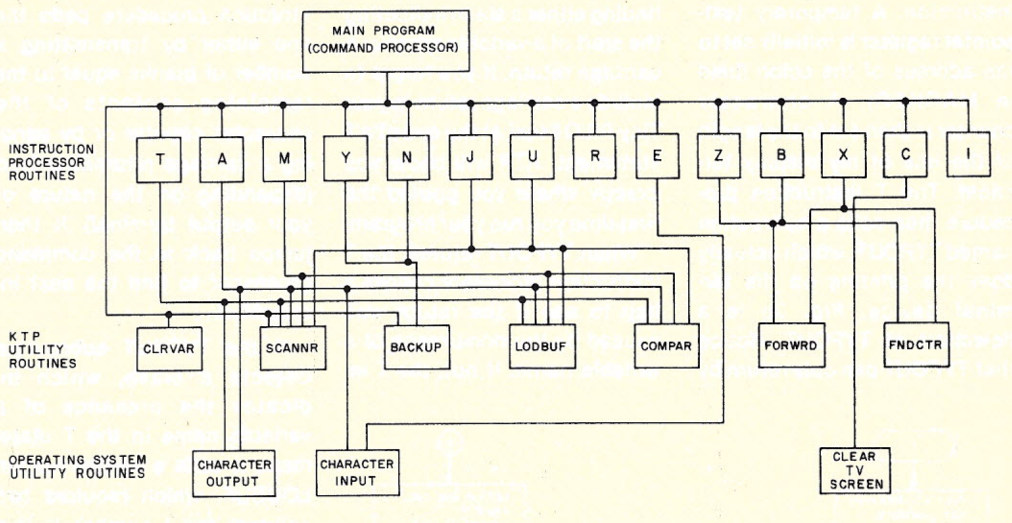



*Fig. 1. Overall flowchart.*

The first non-blank character preceding the colon is the instruction. The command processor therefore calls a routine named BACKUP, which finds that character by decrementing the pointer until it finds any non-blank character. (I first saw this rather elegant method of finding PILOT commands in an experimental PILOT interpreter by Dean Brown of Zilog, Inc. See reference 2.)

The command processor must now decode the command and jump to the appropriate instruction processor routine. I implemented this function by setting up a jump table. I stored the 14 Tiny PILOT instructions in a table with memory running from location $0DD0_{16}$ to location $0DDD_{16}$. The low bytes of the address of each instruction processor routine were listed *in the same order* from $0DE0_{16}$ to $0DED_{16}$. The high bytes were listed from $0DF0_{16}$ to $0DFD_{16}$. When the command processor finds a match between the candidate instruction and a member of the instruction set, it adds 10H twice to fetch the address of the instruction processor.

Two steps are included in the command processor to make KTP easier to use. First, KTP assumes that the instruction is a T if the search of the instruction list is unsuccessful. Second, the option of typing instructions in lowercase can be implemented by adding a two-byte instruction step immediately preceding the search of the command table that logically ANDs the candidate instruction with $DF_{16}$. This turns lowercase ASCII into uppercase.

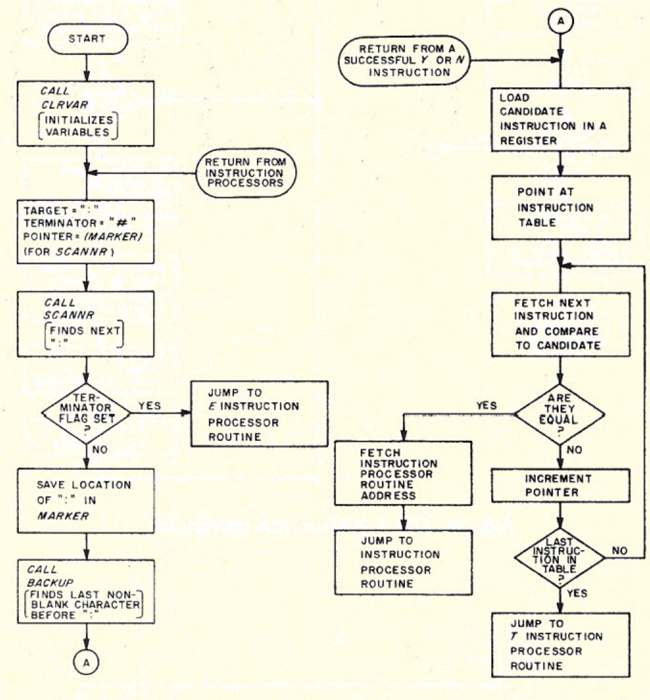



*Fig. 2. KTP main program.*

### The T Instruction Processor

Fig. 3a is a flowchart of the procedure for executing the T

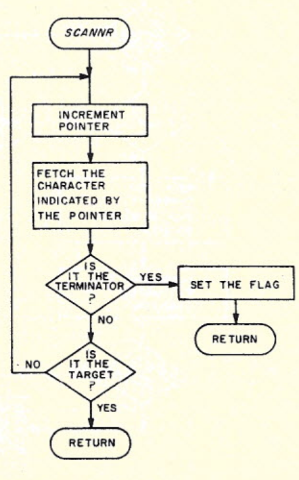



*Fig. 2b. SCANNR.*

instruction. A temporary text-pointer register is initially set to the address of the colon (held in MARKER). A character counter is then set to the length of the line of my display terminal. The T instruction procedure then calls a subroutine named TYPOUT, which actually does the printing on the terminal device. Fig. 3b is a flowchart of TYPOUT. Notice that TYPOUT can only return by finding either a slash indicating the start of a variable name or a carriage return. If you forget to include a carriage return in your Tiny PILOT text at the end of a T statement, KTP will show you exactly where you goofed the first time you run your program.

When TYPOUT returns, the T instruction processor checks a flag to see if the return was caused by the occurrence of a variable name. If not, the T instruction procedure pads the line either by transmitting a number of blanks equal to the remaining contents of the character counter or by sending a carriage return/line feed (depending on the nature of your output terminal). It then jumps back to the command processor to find the next instruction.

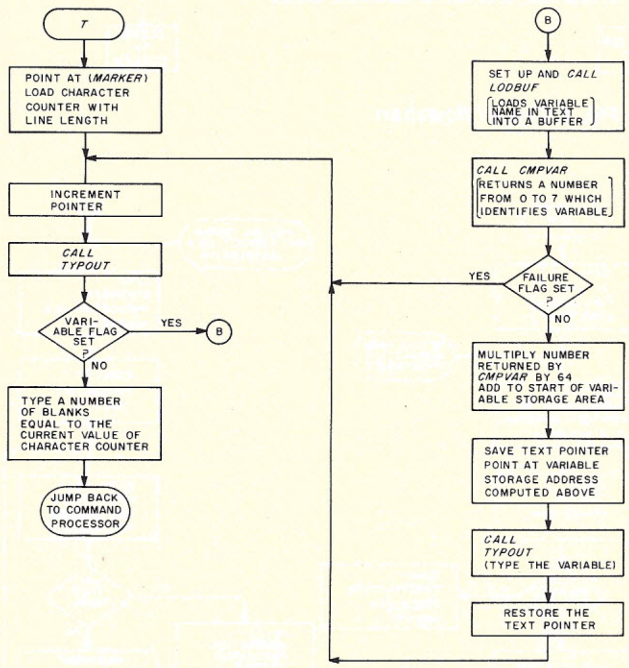



*Fig. 3a. The T instruction processor.*

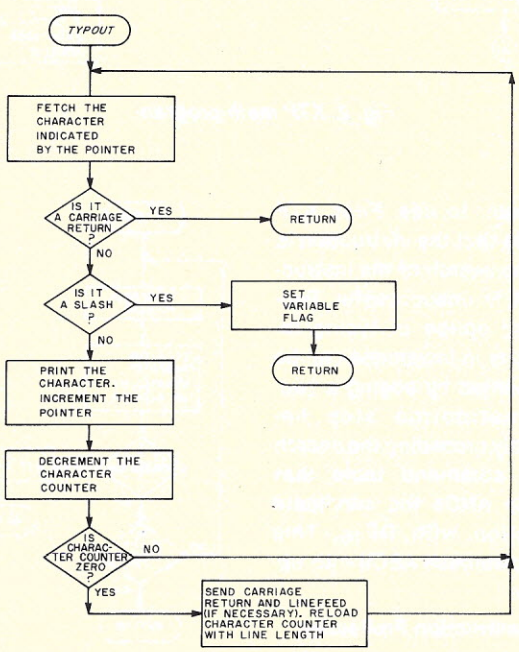



*Fig. 3b. TYPOUT.*

If the TYPOUT subroutine detects a slash, which indicates the presence of a variable name in the T statement, it calls a routine named LODBUF, which requires two pointers and a number. In this case, the *transmitter* pointer contains the address of the first character after the slash starting the variable name in the operand field of the T statement. The *receiver* pointer is aimed at a special buffer area used by the comparison routine. The number (six, in this case) is the maximum length of the character string to be transferred. If LODBUF encounters a second slash before the sixth character, it returns at that point.

The next subroutine, CMPVAR, is actually a special calling sequence for a general string-comparison routine named COMPAR. CMPVAR compares the character string in the comparison buffer with the names of the eight (or less) variables defined by previous A statements. CMPVAR returns a number from 0 to 7, which represents the name of the successfully matched variable. If no match is found, a failure flag is raised. A model of the COMPAR routine can be found in *Scelbi's "8080" Software Gourmet Guide & Cookbook* (see reference 3).

Before calling COMPAR, CMPVAR examines the number of active variables read in by A statements. If zero, CMPVAR immediately sets the failure flag and returns. Obviously, COMPAR will never find the appropriate variable name if there are no variable names defined. If there are variables to examine, CMPVAR points at the first character of the first variable name and calls COMPAR, which compares this character with the first character in the buffer.

If the two don't match, it proceeds to the next variable name, identified by means of the terminal slash ending each variable name. If the first characters match, COMPAR checks the second character in the name against the second character in the buffer. COMPAR continues making comparisons until it either exhausts the list of active variables or succeeds in completing a match of an entire variable name (signaled by matching the terminal slashes).

If CMPVAR returns with the failure flag set, the T instruction processor continues to type with the character in the T statement immediately following the undefined variable name. If CMPVAR has successfully matched the variable name, the T instruction processor computes the address of the variable text storage area by multiplying the number of the variable by 64 (shift left 6 bits) and adding the result to the starting address of the variable storage area.

After the variable text has been typed, the remainder of the operand field of the T statement is typed. The last line is padded and control is returned to the command processor.

**The A Instruction Processor**

Fig. 4 is a flowchart of the A instruction processor. Conceptually, the processor performs six tasks: (1) It clears the 64 character input buffer. (2) It fills the input buffer from the keyboard. (3) It checks the operand field of the A statement to see if the contents of the input buffer are to be stored as a variable. (4) If so, it compares the variable name to those already in use to see if an old variable is being refilled. (5) If necessary, it stores the new variable name and increments the number of active variables. (6) It stores the contents of the input buffer in the first free 64-byte block of the variable text storage area.

After clearing the input buf-

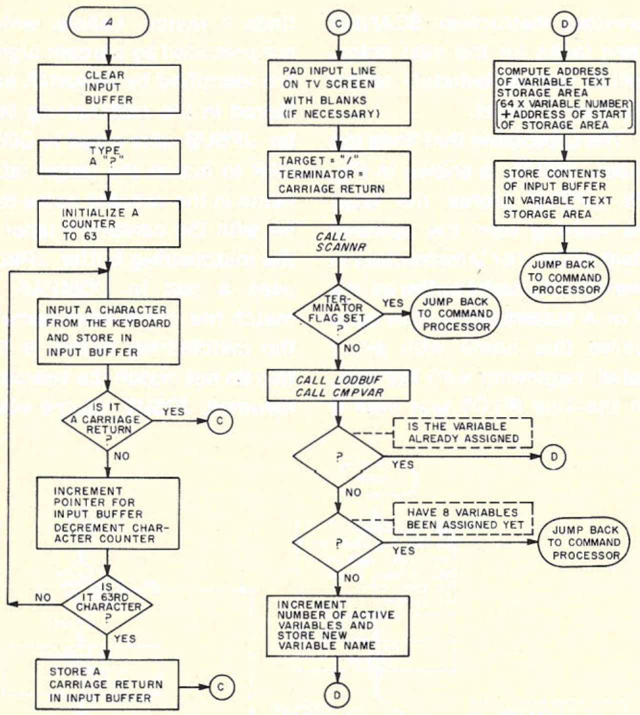



Fig. 4. The A instruction processor.

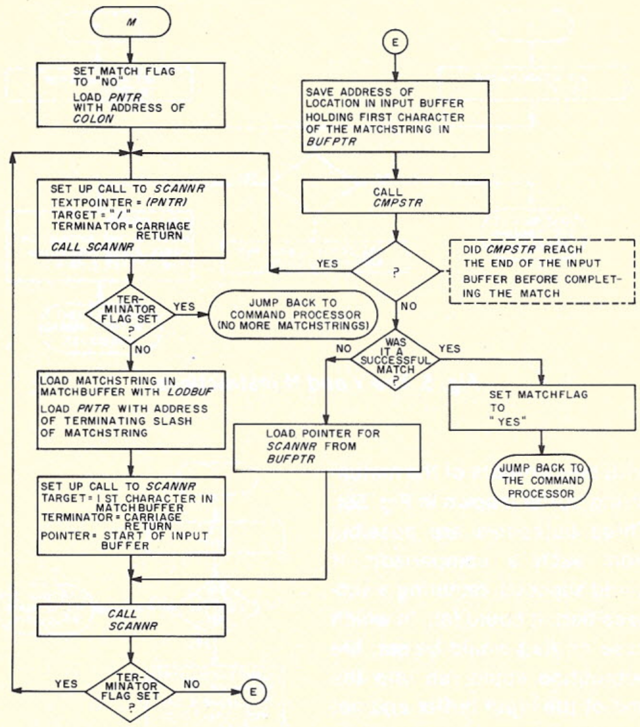



Fig. 5a. The M instruction processor.

fer, the A instruction processor types ? as a prompt character on the screen. The system input routine is then used to read up to 63 characters terminated by a carriage return from the keyboard. If no carriage return has been typed after the sixty-third character, one is automatically entered in the sixty-fourth byte of the input buffer.

The A instruction processor then uses the SCANNR routine to search the operand field of the A statement for a slash. If a carriage return (implying the end of the A statement) is found first, the A instruction processor returns control to the command processor. If a slash is found, the A statement processor tests for an old variable name in exactly the same way as the T instruction processor. If the variable has already been defined, the program skips ahead to compute the storage address and store the text.

If the variable name is new, the A instruction processor tests the number of defined variables to make sure that there is room for another. If so, the number of variables is incremented and the new variable name is stored. If not, the A instruction processor terminates and control is returned to the command processor.

The A instruction processor computes the address of the storage area for the variable text from either the number returned by CMPVAR or the new value of the number of defined variables. The calculation is identical to that in the T processor. The text is then moved from the input buffer to the storage area with a block-move procedure, a single instruction on a Z-80 or a short subroutine (from reference 3) on other chips.

**Processing the M statement**

The M instruction processor is shown in Fig. 5a. It begins by clearing the matchflag, then loads the first matchstring in the operand field of the statement into a buffer storage area. A subroutine call to COMPAR through CMPSTR then matches the contents of the matchstring buffer with the contents of the input buffer. If no match is found, the next matchstring is loaded into the buffer storage area and compared to the input buffer. The M instruction processor repeats this sequence until either a match is found or it runs out of strings in the operand field of the M statement. If a successful match is found, the processor routine raises the matchflag before returning to the command processor.

The M instruction processor uses two pointers. PNTR, by pointing at the terminating slash of the string in progress, keeps track of the matchstrings already tested. BUFPTR points at the last address in the input buffer found to hold the first character in the matchstring and, thus, keeps track of the amount of the input buffer scanned for a match with the current string.

CMPSTR makes a character-for-character comparison of the contents of the input buffer

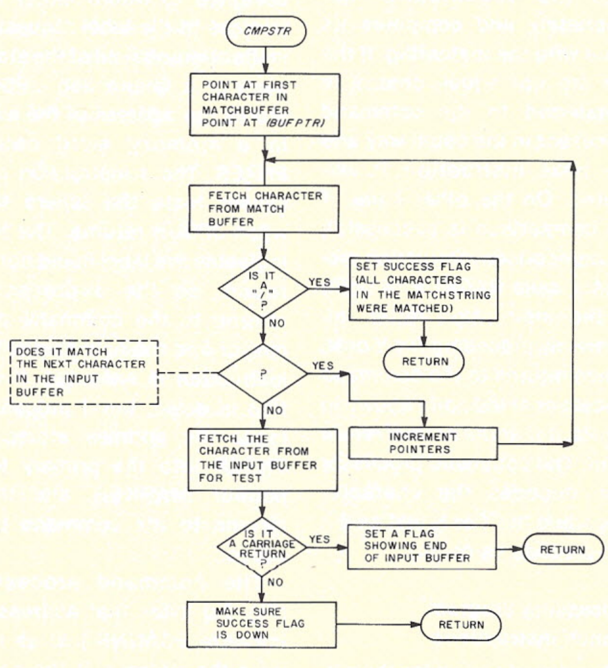



Fig. 5b. CMPSTR.

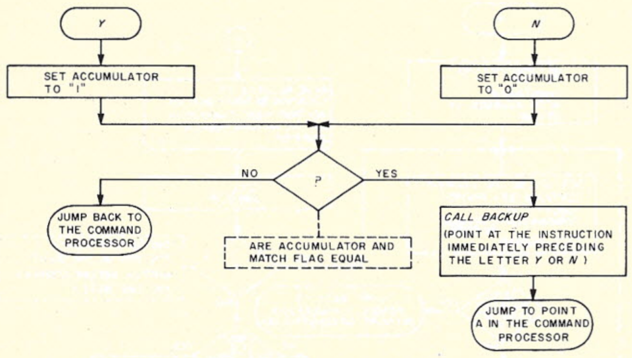



*Fig. 6. The Y and N instructions.*

with the contents of the matchstring buffer (shown in Fig. 5b). Three outcomes are possible from such a comparison: It could succeed, requiring a success flag; it could fail, in which case no flag would be set; the subroutine could run into the end of the input buffer and not match any characters (a flag is needed for this condition). The M statement processor will do something different (go back for a new matchstring) if CMPSTR fails in this way.

### The Y and N Instructions

The routine that processes the Y and N instructions to test the matchflag is very simple (see Fig. 6). The memory byte that holds the matchflag will equal one if the matchflag is set to "yes," and zero if the flag is set to "no." The processor routine sets the accumulator appropriately and compares its value with the matchflag. If the two are not equal, control is transferred to the command processor in the usual way and the next instruction is executed. On the other hand, if the comparison is successful, the comparison instruction processor calls BACKUP to point at the instruction letter immediately preceding the Y or N. It then returns to the command processor at the point shown in Fig. 2a, *not at the normal return point*. The command processor then decodes the character preceding the Y or N just as if it had preceded a colon.

### Processing Program Branch Instructions

Statements involving the program branch instructions J and

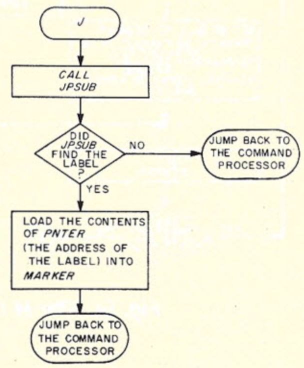



*Fig. 7a. The J instruction processor.*

U are processed in a similar way. Those tasks common to both instruction processors are contained within a subroutine named JPSUB called by both instruction routines.

Fig. 7a shows the procedure for the J instruction. JPSUB is designed to return either the address of the label requested in the operand field of the statement or a failure flag. JPSUB stores the address of the label in a memory word called PNTER. The J instruction processor tests the failure flag when JPSUB returns. The flag indicates the label could not be found, so the J processor returns to the command processor and the next sequential instruction is executed. If the flag is down, the J processor loads the address stored in PNTER into the primary text pointer, MARKER, and then returns to the command processor.

The command processor (Fig. 2a) loads that address to initialize SCANNR just as if it were the address of the colon starting the operand field of the previous instruction. SCANNR then looks for the next colon, which will immediately follow the desired label.

The subroutine that finds the label, JPSUB, is shown in Fig. 7b. JPSUB stores the label namestring from the operand field of the J or U instruction in the variable name buffer as in a T or A statement. It then compares this name with every label, beginning with the first, in the Tiny PILOT text until it finds a match. Labels, which are preceded by percent signs, are identified by SCANNR and stored in the matchstring buffer. JPSUB uses a call to COMPAR to match the target label name in the variable name buffer with the candidate label in the matchstring buffer. JPSUB uses a call to COMPAR to match the target label name in the matchstring buffer. If the two do not match the search is resumed. JPSUB returns when

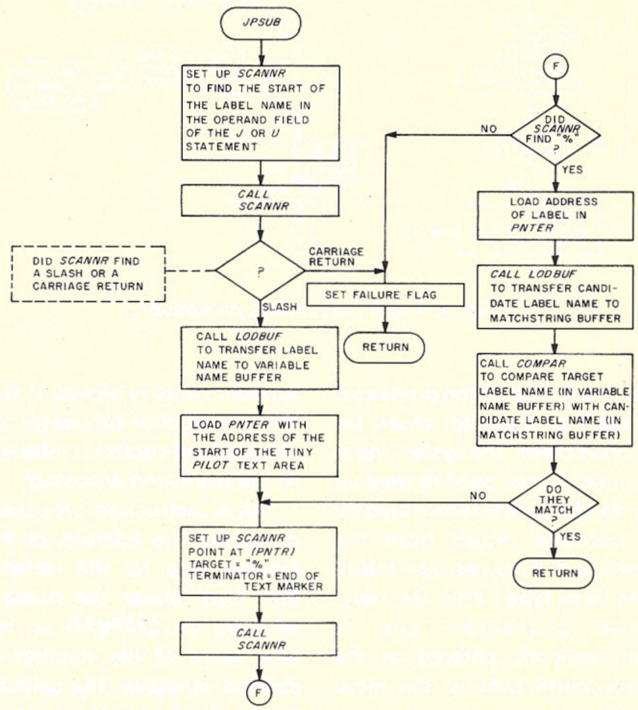



*Fig. 7b. JPSUB.*

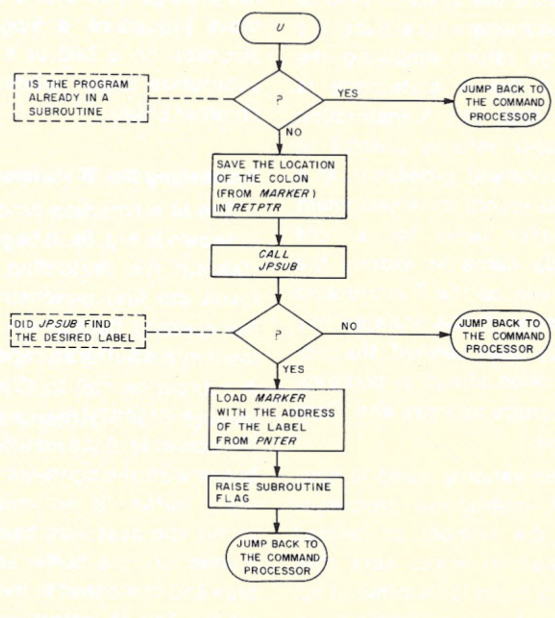



*Fig. 8. The U instruction processor.*

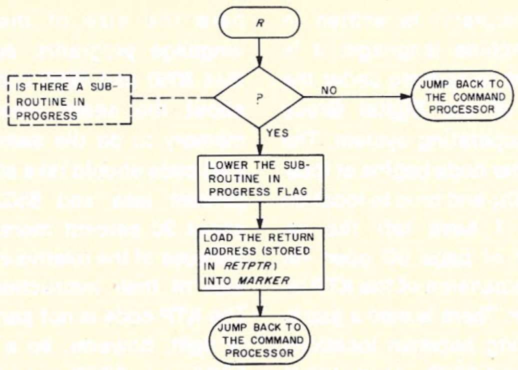



*Fig. 9. The R instruction processor.*

it either finds a successful match or encounters the end of text marker terminating the Tiny PILOT text.

Fig. 8 shows a flowchart for the U instruction processor. The actual branch to the Tiny PILOT subroutine is handled in exactly the same manner as in the J instruction processor. KTP saves the address from which the subroutine was called for the R instruction processor in a two-byte memory word named RETPTR.

KTP allows only one subroutine to be active at a time. This restriction eliminates the need to implement a software stack to hold the return addresses of nested subroutines. Therefore, the U statement processor must check a flag to make sure the programmer is not attempting to call one subroutine from another. Otherwise, the return address of the first subroutine would be lost. If the subroutine flag is raised, KTP will ignore the U statement and execute the next sequential instruction.

The R instruction processor is shown in Fig. 9. First, the R processor examines the subroutine flag to check the validity of the address stored in RETPTR. If no subroutine is active, the R instruction is ignored. If the subroutine flag is raised, the R instruction processor turns it off. It then loads MARKER with the address stored in RETPTR. This is the address of the colon in the U statement that initiated the subroutine. Thus, when the R statement jumps back to the command processor, execution continues with the statement following the subroutine call.

### The Counter Manipulation Routines

Flowcharts for the two instructions that change the values of the four integer counters are shown in Fig. 10a. The two routines are identical until the final step, where the Z instruction sets the counter to zero, while the B instruction "bumps" the counter by adding one to its contents. The majority of the work for both instructions is therefore performed by a subroutine named FNDCTR (see Fig. 10b).

FNDCTR starts with a routine named FORWRD to find the first non-blank character in the operand field of the statement. FORWRD is an exact reverse of the subroutine BACKUP used in the command processor. If the character that's returned by FORWRD is anything other than I, J, K or L, FNDCTR raises an error flag (the carry bit again) and returns. If the character is a valid counter name, FNDCTR subtracts the ASCII code for I. The result, a number between zero and three, is added to the address of the I counter to give the address of the counter requested in the statement.

The contents of the numerical counters are tested by the X instruction processor (Fig. 11). The X instruction processor switches the global matchflag to "no." It must then decipher and evaluate the mathematical expression in the operand field of the X statement. If the expression is true, the X processor will set the matchflag. If false, the matchflag will be left at "no."

The mathematical expression in the operand field is composed of three parts: the name of the numerical counter (I, J, K or L), the symbol for one of the three relational operators (<, =, or >) and the decimal integer constant (a one-to-three digit integer between 0 and 255). The X statement processor uses FNDCTR to obtain the desired counter address, which is stored temporarily in the pointer storage word BUFPTR. A call to FORWRD then fetches the next non-blank character in the operand field. If it is not the symbol for a relational operator, the X processor returns control to the command processor. Otherwise, the X processor subtracts the value of the ASCII code for = from the character for the desired relational operator. The result is −1 for <, 0 for = and +1 for >. This value is temporarily stored, and the X processor again uses FORWRD to find the first character of the numerical constant.

The conversion of numerical constants for ASCII decimal to internal binary is a straightforward process. Many monitors are equipped to do this; if yours is not, you will have to write your own by fetching a character. To make sure it is a number, compare it to the ASCII characters for 0 and 9. Strip the ASCII code by performing a logical AND with $0F_{16}$. Store the result temporarily and fetch the next character. If it is not a number, you're done; the binary number in temporary storage is the desired result. Otherwise, multiply the number in tem-

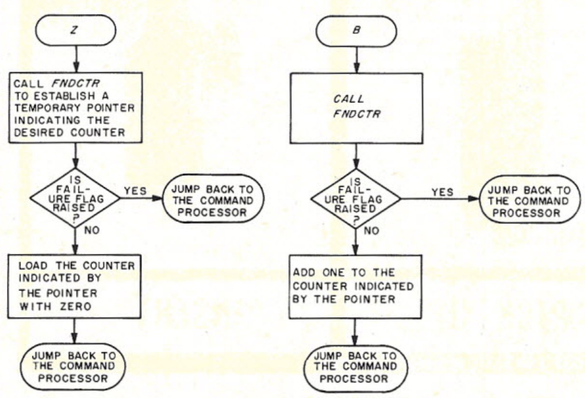



*Fig. 10a. The Z and B instruction processors.*

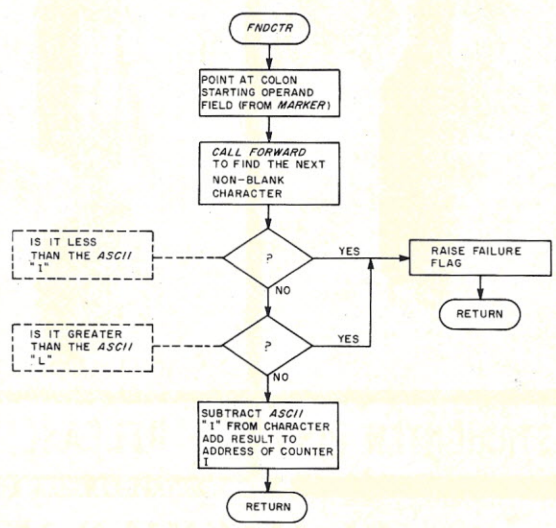



*Fig. 10b. The subroutine FNDCTR.*

porary storage by ten and add the new number.

Multiplication by 10 consists of two left shifts (multiplies by 4), an addition of the original number (results in multiplication by five) and another left shift. Store this result and fetch the third character. If it is a number, repeat the multiply and add steps. Be sure to check for overflow at each step in the third iteration. Three-digit numbers can be as large as 999, but a single byte will only hold numbers less than 256.

The X processor compares the numerical constant with the value in the counter. If the counter value is larger than the constant, the processor subtracts one from the indicator stored for the value of the relational operator. If the constant is larger than the value in the counter, it adds one to the indicator. If they are equal, the indicator is unchanged. The X processor then tests the value of the indicator. If it is zero, the mathematical expression is true. The matchflag is set to "yes" before the X statement processor returns.

#### Miscellaneous Instructions

The E statement processor consists of two calls to system utility routines. The first is a call to the keyboard input routine (simply to throw the computer into a do-nothing loop while the user reads the last text typed on the CRT screen). Tapping the space bar or any other key results in a transfer to the monitor.

The processor for the I instruction is trivial, simply a transfer back to the command processor. Thus, the contents of the operand field of the I statement are ignored. The processor for the C instruction is almost as easy. It is a call to the operating-system utility routine that clears the CRT.

#### Memory Allocation

Obviously, the length of the Tiny PILOT interpreter depends on you and the instruction set for your microprocessor. My KTP interpreter is written in Z-80 machine language; it is designed to operate under the control of the Digital Group TVCOS operating system. The interpreter code begins at location $0880_H$ and runs to location $0C5A_H$. I have left the remainder of page 0C open for future expansion of the KTP interpreter. There is also a gap in the coding between locations $0A67_H$ and $0B00_H$ for possible patching if any bugs turned up after the system was in use. (None have turned up so far.) I use page $0D_H$ as storage for pointers, buffers, counters and variable names. Pages $0E_H$ and $0F_H$ make up the 512-byte storage area required for the eight variables.

I assigned 6K of memory (all I had left at the time) to the Tiny PILOT text area. Even though I have more memory now, I have never felt the need to expand the text area. (An elementary-school child just won't pay attention for much longer than it takes to run a 6K Tiny PILOT program.)

As a rule of thumb to compare the size of machine-language programs, consider that 8080 and 6800 code take about the same amount of memory to do the same task. Z-80 code should take about 20 percent less and 6502 code about 20 percent more space because of the relative efficiencies of their instruction sets. The KTP code is not particularly tight, however, so a skilled 8080 or 6800 assembly-language programmer could fit a Tiny PILOT interpreter into about the same amount of space that I used. The speed of the interpreter should be sufficient to prevent any time delay in executing a Tiny PILOT program.

#### Summary

People who like doing things the easy way can purchase a copy of KTP on a Digital Group format cassette for $15 from Computer Mart, Inc., 1097 Lexington Street, Waltham MA 02154. However you do it, though, get Tiny PILOT up on your system. This is a program that really *is* fun for the whole family, and it just might help some school grades, too. ■

### References


1. *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia,* Vol. 1, No. 1, January 1976. (This is a copy of the original description of the interpreter for Tiny BASIC. A good starting place for those who know little about interpreters.)

2. "Z-80 PILOT An Experimental Version"—program by Dean Brown, comments by Marc LeBrun, *People's Computer Company,* Vol. 5, No. 5, March-April 1977, p.2. (This is a description and assembly-language listing of an experimental interpreter written by Dean Brown of Zilog, Inc. It has a rather spectacular bug, which is explained in their article. I learned a lot by studying this article, but then decided to start from scratch and design my own interpreter.)

3. *Scelbi's "8080" Software Gourmet Guide and Cookbook* by Robert Findley, Scelbi Computer Consulting, Inc., Milford CT. (This book will teach you how to write all the machine-language-dependent utility routines you'll need for a Tiny PILOT interpreter or other projects. A corresponding volume has also been written for the 6800. Owners of other CPU chips can only hope that the Scelbi people get around to their chip.)

4. "Source Code for 8080 PILOT, Version 1.1" by John Starkweather, *DDJ*, etc., Vol. 2, No.5. (This interpreter is about twice as long as KTP. It is less suitable for children, but still is not hard to use.)

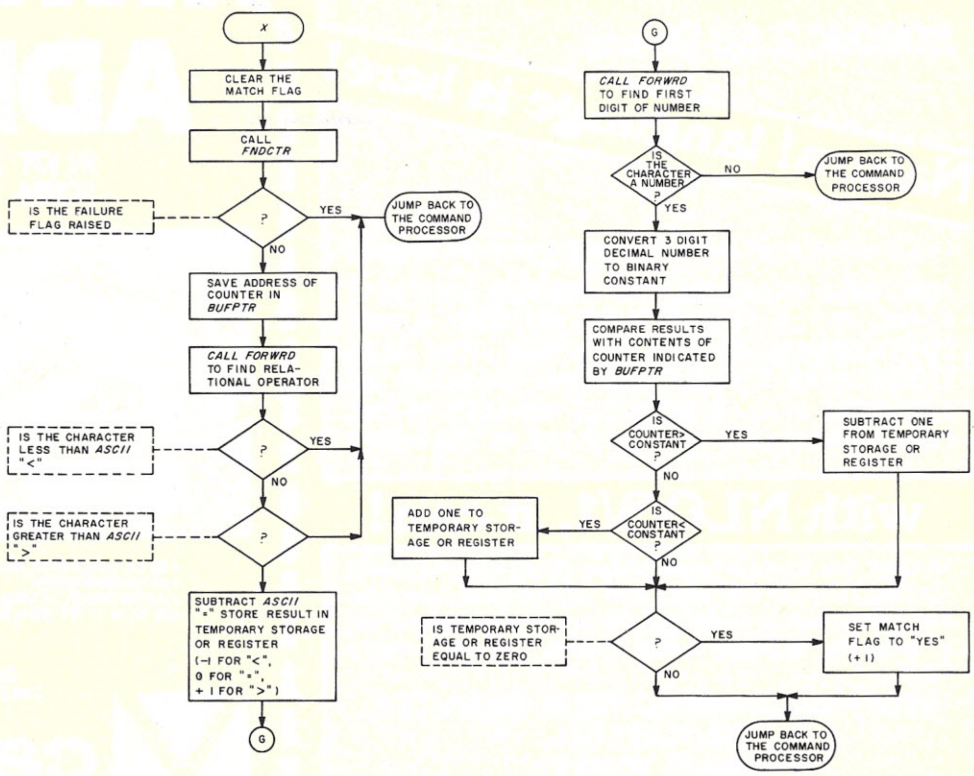




*Fig. 11. The X instruction processor.*

## 8080 & Z80 Users

# Natural language is here!

*Can your computer read and solve this problem by itself?*

"ON THEIR VACATION, TOM AND DICK VISITED A FARM. WHILE THERE, THEY NOTICED A PEN CONTAINING CHICKENS AND PIGS. TOM SAID THERE WERE 3 TIMES AS MANY CHICKENS AS PIGS. DICK SAID HE COUNTED 100 LEGS IN THE PEN. HOW MANY CHICKENS WERE IN THE PEN?"

## with NLOS/1, it can!

NLOS/1 is a cassette-based system requiring a minimum of 12K, a serial I/O board and any cassette interface. The system comes complete with a fully documented set of assembly language source listings. The cost is only $200.

**STOP PROGRAMMING YOUR COMPUTER, EDUCATE IT! ORDER TODAY!**

**CYBERMATE** C57
R.D. #3 BOX 192A
NAZARETH PA 18064

# ADM-3A

IN KIT FORM $ 756 00*

- All equipment F.O.B. Dallas, Texas or Anaheim, California
- Terms — check or money order with order
- Delivery in 30 days
- Add $20 for Shipping and Handling

**OPTIONS:**

- Lower case option add $100.00
- Numeric key pad option add $75.00

- 80 CHARACTERS/LINE
- 24 LINES/SCREEN
- ADDRESSABLE CURSUR
- 9, 10, or 11 BIT WORDS
- 75-19,200 BAUD
- FULL & HALF DUPLEX
- ODD/EVEN/NO PARITY
- RS232 INTERFACE OR 20 ma CURRENT LOOP

GET COMPLETE DETAILS WITH A DIRECT CALL:
214 258-2414 TWX 910-860-5761 TELEX 73-0022
800 527-3248

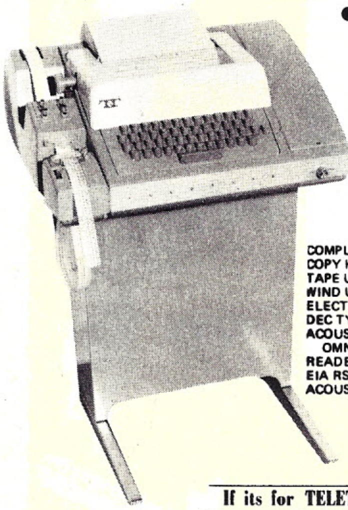
**capital** equipment brokers C53

930 N. BELTLINE • IRVING, TEXAS 75061

# TELETYPE® MODEL 33 ASR

**COMPUTER I/O COMPLETE WITH:**

- Tape Punch
- Ready to Go
- Tape Reader
- Line/Local Wired
- Guaranteed 30 Days



## $840.00

INCLUDING PACKING
F.O.B. - N.J. FACTORY

| | |
|---|---|
| COMPLETE COVER READY TO INSTALL | $ 35.00 |
| COPY HOLDER (182036) | $ 14.00 |
| TAPE UNWINDER (12" NEW) | $ 55.00 |
| WIND UP PAPER TAPE WINDER (1") | $ 52.00 |
| ELECTRIC PAPER WINDER (LPW-300 NEW) | $ 50.00 |
| DEC TYPE READER RUNCARD | $ 45.00 |
| ACOUSTIC COUPLER OMNITEK 501A (NEW) | $160.00 |
| READER POWER PACK (182134) | $ 35.00 |
| EIA RS-232 INTERFACE | $ 55.00 |
| ACOUSTIC COVERS with FAN (NEW) | $258.00 |

**If its for TELETYPE® We Have It.**

**If you don't see what you need, CALL or WRITE!**

**TELETYPEWRITER COMMUNICATIONS SPECIALISTS**
550 Springfield Avenue • Berkeley Heights, N. J. 07922
(201) 464-5310 • TWX: 710-986-3016 • TELEX: 13-6479 T13

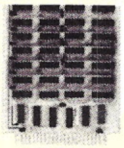
*SUBSIDIARY OF VAN'S W2DLT ELECTRONICS.*

**DO THESE NUMBERS MEAN ANYTHING TO YOU?**

**44 PINS**
**4.5" x 6.5" CARDS**
**5 VOLTS**

Wasatch delivers quality system components from stock

| | Assem./Tested | Kit | PCB |
|---|---|---|---|
| • 4K x 8 RAM | $89.95 | $79.95 | $22.50 |
| • 2K/4K x 8 PROM | | 79.95/129.95 | 22.50 |
| • 8-SLOT MOTHER | 50.00 | 40.00 | 20.00 |
| • 80 LINE DIGITAL I/O | | 59.95 | 20.00 |



4K RAM

New 5V system components under development

- 8K x 8 EPROM (2716)
- 16/32 Channel, 8 Bit A-D
- 4 Channel, 8 Bit D-A

Wasatch also delivers prime semiconductor memory at reasonable price, in addition to custom design services.

**Wasatch**
SEMICONDUCTOR PRODUCTS

25 SOUTH 300 EAST · SUITE 215 · SALT LAKE CITY, UTAH 84111 · 714/752-1374

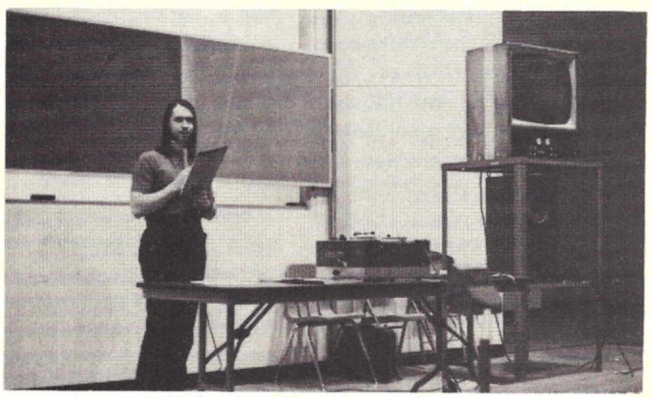
# TCF-78

**TRENTON STATE COLLEGE**
Trenton, New Jersey

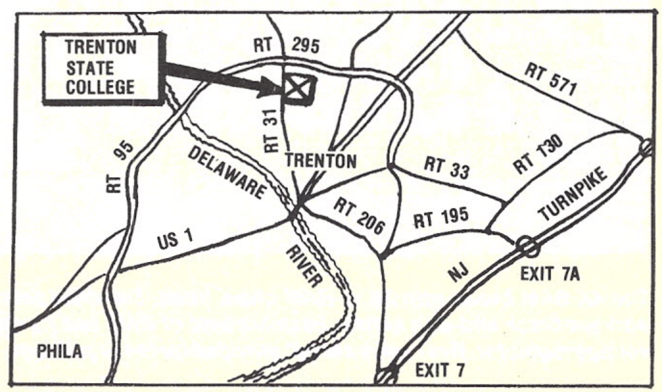
## The Third
# Trenton Computer Festival
## April 22 & 23, 1978

**Super Outdoor Flea Market**
Surplus computer gear, bargains galore, over 5 acres of space ($2/spot, no electricity).

**Indoor Commercial Exhibit Area**
90 exhibitor booths showing newest products; special discounts; funky games to play.



**Forums, Talks & Seminars**
Meet the leading experts and hear sessions on robots, computer music, amateur radio, etc.




**Convenient To NY, PA, MD & DEL**
Easy to get to; free parking for over 6,000 cars.

**Hundreds of Door Prizes**
★ ★ ★
**Banquet Saturday night**

*For more information call:*
609-771-2487 or 201-277-2063

Admission $4 Students $2

*Hours:* 10AM-6PM Saturday; 10AM-4PM Sunday

★ ★ ★

Banquet $9. **Avoid disappointment — preregister for the banquet!** Send your check for $9 per person to: TCF-78, Trenton State College, Trenton, N.J. 08625.

*Sponsored By:*

Amateur Computer Group of New Jersey

Philadelphia Area Computer Society

Trenton State College Digital Computer Society

Dept. of Engineering Technology Trenton State College

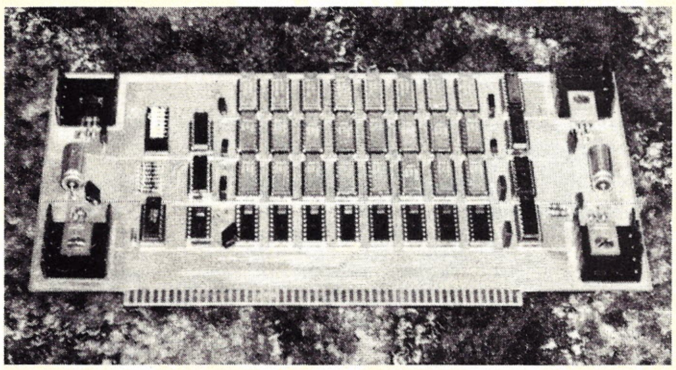
I.E.E.E. Princeton Section

*Joe Kasser*
*11532 Stewart Lane*
*Silver Spring MD 20904*

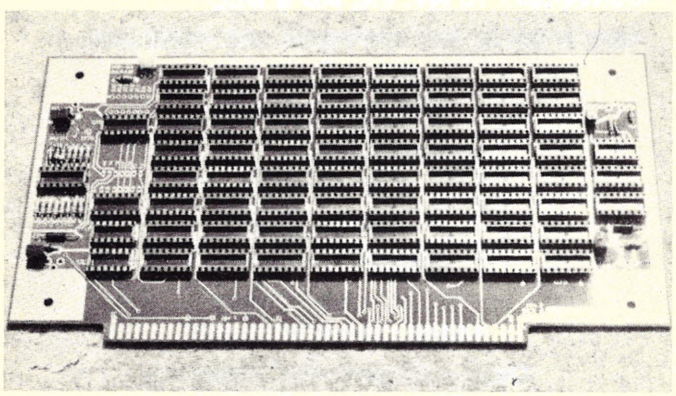
# Blue Is the Color

## *Solid State Music is the company*

Lurking in the back pages of computer hobbyist magazines are some low-key advertisements that describe a growing family of widely distributed, economical Altair-compatible computer cards: Solid State Music's Cybercom boards.



*The 4K RAM board with 3K of RAM chips fitted. Switches select address block and wait states. Each kilobyte of RAM has its own voltage regulator. Heat sinks are not provided for the regulators.*



*A partially assembled 8K RAM board. Sockets are used for every IC. Four regulators are provided for each 2K of memory. Again, no heat sinks are provided. The battery backup bus connector is at the top of the card.*

Each card is an industrial-quality double-sided glass epoxy printed circuit board with plated-through holes (and I shouldn't forget the beautiful blue color) and ample, well-written documentation. For each card, the theory of operation is described, full schematics and sample uses are given, and software is even provided where applicable. The parts supplied with each kit are of good quality. The IC sockets are all low-profile TI types, and the memory and microprocessor support ICs are made by NEC.

Some of the cards not reviewed here, but which are now available, are a music interface card, a 16K static RAM card, a 16K EPROM card using 2708s and a two-serial four-parallel port I/O card. If they are built and documented to the same standards as the kits reviewed here they should also be good values.

Now, let's consider some of the kits currently available from Solid State Music.

### MB-3 2K/4K PROM Board

The MB-3 will handle 16 1702-type EPROMs. Heat sinks are not included, and I suggest that you add your own. The −9 V supply is obtained by floating a three-terminal −8 V regulator. Since all the 1702As I've tried work on −8 V, it is possible that the floating components could be eliminated. But this is a review, not a redesign.

The base address of the card can be set to any 4K block, and up to four wait states can be set up for slow PROMs. All such preset adjustments are made by DIP switches.

### MB-4 4K/8K RAM Board

While this card has space for 4K of 21L02-type RAM, 8K can be put on the card by piggybacking an additional 4K on top of the sockets. The board was laid out with that feature in mind, and the decoding for the second 4K chip select is available at strategically placed pad locations. The documentation gives full instructions on how to carry out the operation. There is a memory protect flip-flop on the card to inhibit memory write operations. The wait states are fully synchronous for slow memories, but as supplied, the kit requires zero wait states. If you already own a set of slower 2102s you can buy the blank board and use your chips. The base address of the card can be set to any 4K block between 0000 and F000 hex. All presetting is done by means of DIP switches. The instruction set shows how to include the Phantom Pulse for the Processor Technology SOL.

**MB-6 8K RAM Board**

This card contains 8K of 21L02 RAM, and the base address can be set to any 8K value. Up to two wait states can be preset. A memory protect feature enables blocks of memory to be protected in sets of 256, 512, 1K, 2K, 4K and 8K bytes. Preset adjustments are by DIP switches. All ICs are provided with low-profile sockets. A jumper location and simple modifications are available to implement the Phantom Pulse for the SOL machine. The board is soldermasked to aid in preventing unwanted connections. Four three-terminal regulators are provided without heat sinks, and they do get warm if forced air cooling is not utilized. Backup battery power for the standby mode is available through a connector at the top of the card.

Sockets are not provided for the DIP switches. I would have preferred them so I could replace the switches by a fixed jumper pad on a chip header and use the switch on another card under development; but that is personal preference only. There are only four disk capacitors on the whole card, which seems to be too few, yet the card works beautifully. The memory protect circuit uses a 21L02, but if the feature is not desired, that chip becomes a spare.

**MT-1 Motherboard**

This is a single-sided 3/16th-inch-thick board, predrilled for up to 15 connectors spaced 3/4 inch apart Holes are also drilled to support Altair-style card guides. The holes in the board are large enough for wire-wrap sockets as well as solder tail. The board can be sliced on a guillotine-type cutter into sections for special-purpose equipment. Holes and space are provided for bus-decoupling capacitors on all three power bus lines. A set of holes has been drilled at one end of the board to accommodate an extension unit or line terminators.

**XB-1 Extender Board**

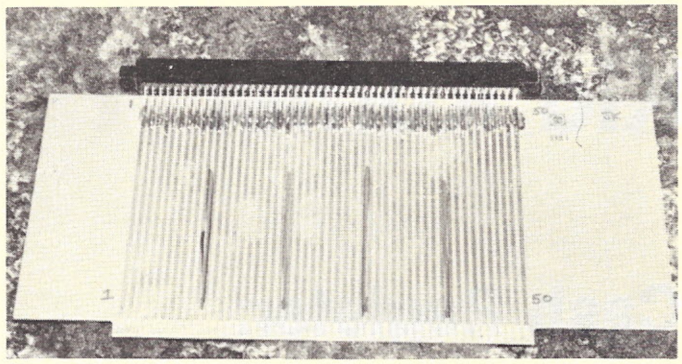
The extender is a no-nonsense, no-frills board that does only what it advertises. It extends all 100 bus lines upward so that a card can be worked on while connected to the bus. It is designed so that a wire-wrap connector can be soldered to the top of the card. If care is taken, the connector pins can be bent outward before soldering, leaving points for attaching a scope probe. This type of card is a must for troubleshooting.

**VB-1 Video Interface**

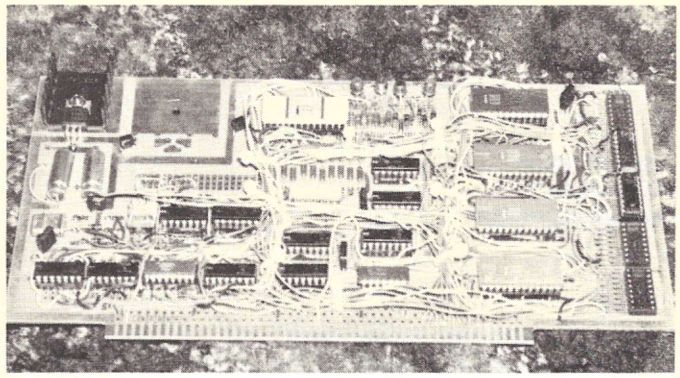
The VB-1 displays uppercase and lowercase graphics and Greek characters using a Motorola MCM 6571 integrated circuit. Sixteen lines of either 32 or 64 characters can be displayed, and the video is available at the top of the card as composite video or as separate video and sync. The characters can be displayed as black on white or white on black, and a mixture of characters and graphics can also be displayed. As the choice between reverse video or graphics is switch-selected, those features cannot be mixed. One kilobyte of on-card memory holds the fully interlaced screen display. The supplied memory is 21L02-4 and does not require any wait states.

Documentation is supplied, and the new software package is excellent. It shows how to use the alphabetical and graphical displays and contains sample programs, including one that lets you address each dot on the screen separately. This feature is great for pong or space-war programs or just plain doodling. The software is set for a base address of 3000 hex. However, if the switch-selected address is set to CC00 hex, Processor Technology software will run (but not scroll) without hardware or software modification.

Control of the display is shared between hardware and software. This may or may not be an advantage in your application. Even the cursor character can be changed under software control. The Greek letter feature is useful to supply prompts in the event of user input errors.



*Extender board with connector fitted at the top. The connector is an Imsai-type with wire-wrapped pins. If the pins are bent before soldering it is possible to make hooks for attaching a test probe.*



*The I/O board kludged for 8251 serial I/O programmable ASCII/Baudot 45.5 at 110 baud. (My kludge—SSM's PC decoding and buffering.) This card contains two input and two output ports feeding a tape reader, keyboard and punch, as well as the 8251 circuitry.*

**IO-2 I/O and Universal Board**

The IO-2 provides one input and one output port plus decoding for nine ports. Output is from a ribbon cable via a DIP socket, and a short length of cable is supplied with the kit. Pads are placed on the board for all manner of DIPs, and space is also provided for two three-terminal voltage regulators, one of which is dedicated to the 5 volt supply.

Schematics and suggestions are provided for adding PROMs or a serial interface for a CRT or Teletype terminal. Instructions are also given to wire the serial port for Mits or Processor Technology software compatibility. There is a schematic for generating multifrequency clocks for a UART from the 2 MHz bus clock (you can get 1200, 600, 300, 150 or 110 baud). The IC socket holes will accommodate either solder-tail or wire-wrap sockets. The IO-2 is an excellent board for general-purpose I/O work.

**Summary**

The cards are well made and can be assembled in one or two evenings. The gold-plated connector even plugs into the socket without any shaving operations. The cards are available assembled, as kits or, in some cases, as blank boards with instructions, which makes them well suited to club projects. They are the best value I've seen yet. About the only fault I could find was that they were not drilled for card pullers.■

# Kid Korner

## Cash Register: a practical math simulation

*John Eric Victor*
*11 Idar Court*
*Greenwich CT 06830*

*Learning math can be fun with the author's Cash Register game, which simulates a checkout counter at a grocery store. The program is written for the TRS-80.*

A recent testing of high-school seniors in Dade County, Florida, revealed that almost half did not possess the minimum basic math skills to fill out a check or add up a grocery list. Fifty years ago, these students would have had enough reading and math skills to get by . . . today, many of them will be unemployable.

As our society becomes increasingly dependent on complex technology, our present educational system will become less and less able to meet the demands made on it. The system needs substantial changes, but these will not appear overnight. And given the record of some educational reformers, these changes could produce a system even worse than the one we have now. As a practical matter, the best short-range solution might be for parents to take some of the responsibility for their children's education.

For many families a computer may be just what the doctor ordered. At $600, the TRS-80 costs about as much as some of the more expensive sets of encyclopedias. On one level, the computer can test children for necessary basic skills. On another level, the computer can give children drill and practice in skills already learned. The computer can also be used to teach new material in such a way that a program is individualized for each child. Best of all, this can all be done without adult supervision, since the computer acts as the tutor. And the home computer is available whenever the child wants to use it.

One of the problems with traditional teaching methods is that they do not simulate the real world very well. The computer, on the other hand, excels at simulations. Some of the more popular computer games are, in fact, simulations, and many of these programs can make abstract concepts more meaningful to children than straight drill and practice.

This program, written for my new TRS-80, is a simulation of a checkout counter at a grocery store. In the Cash Register program, the customer buys an item costing X and pays for it in whole dollar amounts. The child must then make change in dollars, half dollars, quarters, dimes, nickels or pennies. Change can be given in any denomination as long as the total amount is correct (as would be the case in real life).

For example, an item costs 50 cents and the customer pays one dollar. The child might then type in one half dollar or 50 pennies for change, which would be considered correct since either one adds up to 50 cents. The child then gets a CORRECT CHANGE! THANK YOU, and the program goes to the next situation. If the change was incorrect, the child is told whether the amount was too little or too much and by what amount, and the problem is repeated. An incorrect entry can be changed by typing in a −1 and starting the problem over again. For those who feel this may not be a practical way to learn how to make change, I remind you that it is usually taught out of a book (in school)—and the computer is more interactive than a book!

The situations are stored in DATA statements so the problems can be presented in order of difficulty. The program user can also add more problems with additional DATA lines. Line 5000 contains a flag that restores the DATA to the first situation.

### Working with Radio Shack Level I BASIC

The TRS-80 is supplied initially with Level I BASIC in 4K of ROM. This BASIC has its limitations like any 4K BASIC, but it still has some great features for computer-assisted instruction. One of these is the clear screen command (CLS), which allows the programmer to remove distractions and old problems from the screen. The PRINT AT command allows the programmer to position text and the cursor anywhere on the video screen. These two commands can eliminate some of the annoyance caused by scrolling, as well as provide the programmer with special effects such as animation.

This BASIC also recognizes string variables that are restricted to A$ and B$. Neither can be longer than 16 characters, and they cannot appear as part of a logic statement (i.e., IF A$ = "YES" THEN GOTO 50).

There was one problem I did have with this BASIC. The present form of the Cash Register program uses integer math instead of decimals. When I first wrote the program, I used decimal coefficients instead of whole numbers, but I could not get the IF-THEN statements to work properly. Even when I input the correct answer, the program would indicate that my answer was too high or too low by an amount such as 1.9 E −07! Obviously, there are some inaccuracies in the floating-point arithmetic.

When adding more DATA lines to this program, the user must be careful to get the data in the correct order. The program is written so that the first piece of data read is a string for A$—the name of the item being purchased. The next piece of data is the price in the form of a string. I did it this way so the price could be reported in a variety of forms such as $1.50, or a dollar and a half, etc. The next piece of data is the price of the item in the form of an integer or whole number, and the last piece of data is the amount paid (also an integer). Another way to look at this is to think of all of the numbers in cents rather than dollars.

For the sake of clarity, I did not abbreviate any of the program statements. However, the reader may find program entry easier and faster if abbreviations, such as P for PRINT or IN for INPUT, etc., are used. In the case of direct execution in an IF-THEN statement, the THEN is optional. For example, IF A=1 THEN CLS can also be written IF A=1 CLS. The program in unabbreviated form takes up 2645 bytes of memory.

**Converting to Other BASICs**

Converting this program to other BASICs should not be too difficult. The first thing that must be done is to drop all of the CLS and PRINT AT statements. The coin denominations in lines 350 to 390 can be printed as input prompts:

```
355 INPUT "DOLLARS", S
```

```
10  REM $$ CASH REGISTER $$ BY JOHN ERIC VICTOR
12  REM IN RADIO SHACK LEVEL I BASIC
15  REM INTRO GRAPHICS IN LINES 20 to 60
20  CLS
30  FOR X=1 TO 192 STEP 2
40  PRINT AT X;: PRINT "0": NEXT X
50  FOR X=1 TO 192 STEP 2: PRINT AT X;: PRINT "$": NEXT X
60  FOR X=1 TO 192 STEP 2: PRINT AT X;: PRINT "0": NEXT X
80  REM PROGRAM
100  PRINT "THIS IS THE GAME OF $$ CASH REGISTER $$."
110  PRINT "PRETEND THAT YOU ARE RUNNING THE CASH REGISTER"
120  PRINT "AT A LOCAL GROCERY STORE. I WILL BUY THINGS AND"
130  PRINT "GIVE YOU MONEY. YOU WILL GIVE ME CHANGE. YOU CAN"
140  PRINT "GIVE ME THE CHANGE IN PENNIES, NICKELS, DIMES OR WHATEVER,"
150  PRINT "BUT IT MUST ADD UP TO THE CORRECT TOTAL."
160  PRINT:PRINT "PRESS ENTER TO START . . . ": INPUT A$
200  REM A$= NAME OF ITEM IN DATA, B$= PRICE
205  REM T= PRICE, G= PAYMENT
210  REM S= DOLLARS, H= HALF DOLLARS, Q= QUARTERS, D= DIMES
220  REM N= NICKELS, P= PENNIES
300  REM PRESENT PROBLEM
305  CLS
310  READ A$, B$, T, G
312  IF G= 0 THEN RESTORE: GOTO 310
315  PRINT
320  PRINT: PRINT "I BOUGHT";A$;"THAT COST";B$;"."
325  PRINT "I GIVE YOU";G/100;"DOLLAR(S) FOR IT. WHAT DO YOU GIVE ME"
330  PRINT "IN CHANGE? (START WITH DOLLARS AND WORK DOWN TO PENNIES."
332  PRINT "TYPE -1 TO REDO THE PROBLEM.)"
335  REM 3 SPACES BETWEEN THE HEADINGS
337  PRINT
340  PRINT "DOLLARS   HALF DOLLARS   QUARTERS   DIMES   NICKELS   PENNIES"
345  REM NEXT STATEMENT CLEARS LINE IN CASE OF ERROR
350  PRINT AT 512
353  REM MAKE CURSOR MOVE TO EMPTY COLUMN
355  PRINT AT 512;: INPUT S: IF S= -1 THEN 350
360  PRINT AT 522;: INPUT H: IF H= -1 THEN 350
370  PRINT AT 537;: INPUT Q: IF Q= -1 THEN 350
380  PRINT AT 548;: INPUT D: IF D= -1 THEN 350
385  PRINT AT 556;: INPUT N: IF N= -1 THEN 350
390  PRINT AT 566;: INPUT P: IF P= -1 THEN 350
400  REM CHECK TO SEE IF CHANGE IS CORRECT . . .
410  C= S*100 + H*50 + Q*25 +D*10 + N*5 + P
420  IF G - C = T THEN PRINT "CORRECT CHANGE! THANK YOU.": GOTO 500
430  IF G -C<T THEN PRINT "TOO MUCH. YOU OVERPAID ME BY";
440  IF G - C>T THEN PRINT "NOT ENOUGH. YOU UNDERPAID ME BY";
445  REM DIFFERENCE BETWEEN RIGHT AND WRONG AMOUNTS
450  A= ABS(G - C - T)/100: Z= INT(A)
455  Y= ABS(ABS(G - C - T) - Z*100)
457  IF Z= 0 THEN PRINT Y;"CENTS.": GOTO 465
460  PRINT Z;"DOLLAR(S) AND";Y;"CENTS."
465  PRINT: PRINT "TYPE 0 AND PRESS ENTER.": INPUT A
467  CLS
470  GOTO 315
500  PRINT: PRINT "TYPE 0 TO GO ON, OR TYPE 1 TO STOP. THEN PRESS ENTER."
505  INPUT A
510  IF A= 1 CLS: PRINT AT 448;: PRINT "I HOPE YOU ENJOYED THE GAME.": END
520  GOTO 300
1000  DATA "SOAP", "50 CENTS", 50, 100
1005  DATA "PAPER TOWELS", "75 CENTS", 75, 100
1010  DATA "MILK", "95 CENTS", 95, 100
1015  DATA "A DOZEN EGGS", "92 CENTS", 92, 100
1020  DATA "STEAK", "$1.50", 150, 200
1025  DATA "3 STEAKS", "$3.50", 350, 500
1030  DATA "CAT FOOD", "82 CENTS", 82, 100
1035  DATA "BAG OF FLOUR", "$1.25", 125, 500
1040  DATA "ORANGES", "$1.03", 103, 500
1045  DATA "BAG OF GROCERIES", "$18.07", 1807, 2000
5000  DATA "X", "X", 0, 0
```

*Program listing.*

Another programming change that may be required involves the GOTO statements. In Radio Shack BASIC, when an IF-THEN condition is false, the interpreter goes to the next line. It does not execute any other statements on the line with the IF-THEN statement. In the line shown in Example 1, if A does not equal 1, the remaining statements on the line are not executed.

With some forms of BASIC the interpreter will continue along the line even if the conditional statements are false. If this is the case, the IF-THEN statement can be changed so that on a true condition it sends the computer to a GOSUB routine that contains all of the statements to be executed when the condition is true.

### Testing

Just as a program is run on a computer to debug it, an educational program should be run with human subjects to identify any teaching bugs. All of the programs that appear in this column have been tested to some extent. However, I would like to see some results from testing a cross-section of children from various backgrounds. I would appreciate hearing from *Kilobaud* readers who have tried out the programs on their children. ■

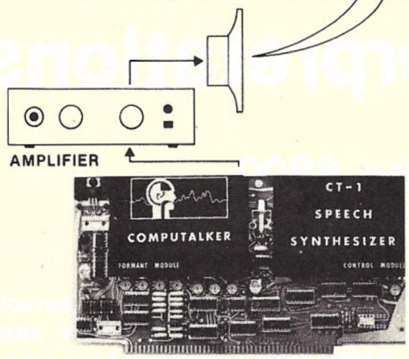
```
510 IF A=1 THEN CLS: PRINT AT 448;: PRINT "I HOPE YOU ENJOYED THE GAME": END
```

*Example 1.*

```
THIS IS THE GAME OF $$ CASH REGISTER $$.
PRETEND THAT YOU ARE RUNNING THE CASH REGISTER
AT A LOCAL GROCERY STORE. I WILL BUY THINGS
AND GIVE YOU MONEY. YOU WILL GIVE ME CHANGE. YOU CAN
GIVE ME CHANGE IN PENNIES, NICKELS, DIMES OR WHATEVER,
BUT YOU MUST GIVE ME THE CORRECT CHANGE.

PRESS ENTER TO START . . .

I BOUGHT SOAP THAT COST 50 CENTS.
I GIVE YOU 1 DOLLAR(S) FOR IT. WHAT DO YOU GIVE ME
IN CHANGE? (START WITH THE DOLLARS AND WORK DOWN TO PENNIES.
TYPE -1 TO REDO THE PROBLEM.)

DOLLARS     HALF DOLLARS    QUARTERS    DIMES    NICKELS    PENNIES
?0          ?1              ?2          ?5       ?10        ?50
TOO MUCH. YOU OVERPAID ME BY 2 DOLLAR(S) AND 0 CENTS.

TYPE 0 AND PRESS ENTER.


I BOUGHT SOAP THAT COST 50 CENTS.
I GIVE YOU 1 DOLLAR(S) FOR IT. WHAT DO YOU GIVE ME
IN CHANGE? (START WITH THE DOLLARS AND WORK DOWN TO PENNIES.
TYPE -1 TO REDO THE PROBLEM.)

DOLLARS     HALF DOLLARS    QUARTERS    DIMES    NICKELS    PENNIES
?0          ?1              ?2          ?-1
?0          ?1              ?0          ?0       ?0         ?0
CORRECT CHANGE! THANK YOU.

TYPE 0 TO GO ON, OR TYPE 1 TO STOP. THEN PRESS ENTER.
?1


I HOPE YOU ENJOYED THE GAME.

READY
```

*Sample run.*

# COMPUTALKER




**S-100 BUS**

CSR 1
SYNTHESIS-BY-RULE
SOFTWARE

**SPEAK "KAAMPYUTAOLKER"**

| | |
|---|---|
| MODEL CT-1 SYNTHESIZER | 395.00 |
| CSR1 SOFTWARE SYSTEM | 35.00 |
| DEMONSTRATION CASSETTE | 2.95 |

CALIF. RESIDENTS ADD 6% SALES TAX

WRITE FOR INFORMATIVE LITERATURE

**COMPUTALKER CONSULTANTS**
**P.O. BOX 1951, DEPT. K, SANTA MONICA, CA 90406**

C36

# H8® & TRS-80® SOFTWARE

| ORDER CODE | DESCRIPTION | MINIMUM SYSTEM | PRICE EACH |
|---|---|---|---|
| HEATHKIT® H8 SOFTWARE | | | |
| | Benton Harbor Basic Games | | |
| H8-BG1 | BIORYTHM, CHOMP, TEASER, BATUM CRAPS, TRAP, 23-MATCHES, NUMBER | BHB-8K | $10.00 |
| H8-BG2 | SNARK, REVERSE, STARS, TAXMAN FIPFOP, SLOTS, HURKLE, MUGWUMP | BHB-8K | $10.00 |
| | Machine Language Games (Front Panel) | | |
| H8-MG1 | Manipulation Games | MEM-4K | $10.00 |
| H8-MG2 | Number Games | MEM-4K | $10.00 |
| H8-MG3 | Casino Games | MEM-4K | $10.00 |
| H8-MG4 | Calculator & Clock | MEM-4K | $10.00 |
| | Machine Language Programs | | |
| H8-CODE | Ham Code Practice Program (Front Panel & Terminal Versions) | MEM-4K TRM-4K | $10.00 |
| H8-KING | Kingdom Game | TRM-4K | $10.00 |
| H8-DISM | Program Disassembler | TRM-4K | $10.00 |
| H8-EDUC | Educator-8080 Teaches machine programming interactively | TRM-4K | $10.00 |
| RADIO SHACK® TRS-80 SOFTWARE | | | |
| | Level One Basic Games and Programs | | |
| R8-BG1 | BIORYTHM, CHOMP, TEASER, BATUM CRAPS, TRAP, 23-MATCHES, NUMBER | L1B-4K | $10.00 |
| R8-BG2 | SNARK, REVERSE, STARS, TAXMAN FIPFOP, SLOTS, HURKLE, MUGWUMP | L1B-4K | $10.00 |

LEGEND: BHB-8K = H8+8K+H9/H36 Uses Benton Harbor Basic
MEM-4K = H8+4K ONLY Uses Front Panel
TRM-4K = H8+4K+H9/H36 Interactive thru Terminal
L1B-4K = TRS-80+4K Uses Level I Basic

Includes: Cassette, Source Listing, and User Documentation
Terms: Check, Money Order, BAC/VISA, Master-Chg - NO CODs!
Foreign orders in US funds, plus $1US/Cassette Postage
Shipment: In sequence as received by UPS or 1st Class Mail
Dealer Inquiries Invited; Program Contributions Solicited
Highest Royalties Paid

Multi-Micro Media Corp., P.O. Box 1025, Arvada, Colo. 80001

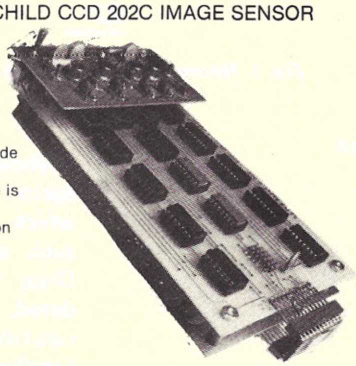
M38

MINATURE SOLID STATE

# 202 VIDEO CAMERA KIT

**FEATURING A... 100 × 100 BIT SELF SCANNING CHARGED COUPLED DEVICE**

THIS UNIQUE UPDATED CAMERA KIT
FEATURES THE FAIRCHILD CCD 202C IMAGE SENSOR



*FEATURES*

- Sensitive to infra red as well as visible light
- May be used in a vacuum, under water and high altitude
- May be used in a magnetic environment because there is no high voltage
- All components mounted on parallel 3¾" × 6½" single sided boards
- Total weight under 1 lb.

*ADVANTAGES*

- IN THE FUTURE WE WILL SUPPLY A COMPUTER VIDEO INTERFACE CARD
- All clock voltages operate at 6V requiring no adjustments
- Higher video output signal
- We supply the power board, so only a 5V 1 Amp power source is needed
- The circuitry has been simplified for easier assembly
- Two level TTL output is supplied for interfacing

**$349.00**

We supply all semiconductors, boards, data sheets, diagrams, resistors, capacitors, and 8MM lens.
Sorry we do not supply the case, batteries and 5V supply.

Add $2.00 Postage and Handling

**SOLID STATE SALES**
PO BOX 74K SOMERVILLE, MASS. 02143 TEL. (617) 547-7053

# NORTH STAR

## DISK ASSEMBLER and DISK EDITOR

Both programs read and write disk files; file size not limited by memory. Assembler will assemble up to ten source files at a time; permits modular programming with programs easily relocated by reassembling at the desired address. Editor does not use line numbers; it searches for strings. Lines may be inserted, deleted and displayed. Large disk source files allow programs to be fully commented.

**ASSEMBLER/EDITOR on disk with users manual...$30**

**COMPUTER SYSTEMS DESIGN**
1611 E. Central Wichita, KS 67214

DEALERS INQUIRIES INVITED

*Gary Gaugler*
*2276 Beaver Valley Rd.*
*Fairborn OH 45324*

# Explained: String Interpretations

## *parsing techniques for the 6800*

Indexing a command input in line for syntax or arguments (parsing) can be a difficult and cumbersome task with the 6800 CPU. Of the few drawbacks of the 6800, the lack of a second 16-bit index register is by far the most prominent. Unlike the architecturally similar 6502, which has two 8-bit index registers, the 6800 has a very powerful 16-bit indexing mode, but for the singular index register. Programmers approach this limitation in many individual ways. Some of the approaches are deceptively risky.

Indexing (indexing is used here to describe many types of string interpretation: tables, user input strings, etc.) is most commonly used when evaluating a human-entered command string for its type, syntax and content. This process is, of course, used in assemblers, editors, compilers and the popular BASICs for the SWTPC 6800. The SWTPC BASIC and co-resident editor/assembler use the stack pointer as a pseudo index register. This is functional but catastrophic in the event of an external interrupt from such sources as a peripheral service request or from a real-time interval timer. The problem centers on the CPU stack's really ceasing to exist since the stack pointer (SP) is loaded with the address of the table or command string to be parsed. When an interrupt would occur, the microprocessor unit (MPU) wants to push the register contents onto the stack (which it does), but the stack actually points somewhere in the user program. Fig. 1 depicts the MPU responding to an interrupt with a normal stack configuration.

When an interrupt is received, the MPU completes execution of the current instruction, then pushes the register contents onto the stack in the following order: conditon code register (CC), B register (B), A register (A), high byte of the index register (X hi), low byte of the index register (X lo), high byte of the program counter (PC hi) and low byte of the program counter (PC lo). This information is used to return to the exact location in the interrupted program after the interrupting device has been serviced. The address of the instruction that would have been executed next if the interrupt had not occurred is the PC hi and PC lo that was pushed onto the stack. It is fatal to the program execution if this return address is lost.

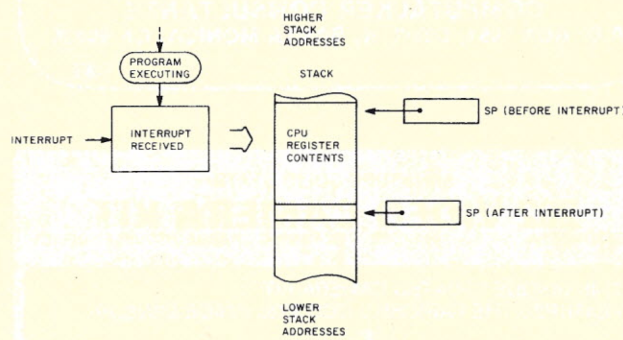



*Fig. 1. Normal 6800 response to an interrupt.*

Now, let's examine how BASIC parses a command string. The command string (that which is typed on the keyboard) must be compared against valid commands to which BASIC will respond, such as LIST, PRINT, etc. Once the command is validated, the address of the routine in BASIC that handles the desired function is loaded into the index register. A zero indexed jump is executed to go to the desired handler. Fig. 2 depicts the MPU organization when performing this function. Program A is a listing of a sample routine that performs the parsing function in the same manner as BASIC.

The stack pointer (a 16-bit register) is used to pull characters from the keyboard in-

*Program A. Sample listing of normal BASIC parsing method.*

```
                       NAM     PARSE
                       OPT     S,NOP
1000                   ORG     $1000
                  *
                  ********************
                  *                  *
                  *    PARSE         *
                  *                  *
                  *********************************************
                  *
                  *  PARSE ROUTINE AS USED IN BASIC, ETC.
                  *
                  *
                  * MAIN PROGRAM CALL TO PARSE SUBROUTINE
1000 BD 2000           JSR     PARSE
                  *
                  *
                  * RETURN HERE AFTER PARSING
                  * IF (X)=LAST ADDRESS OF LOOK-UP TABLE, THEN
                  * KEYBOARD COMMAND CAN'T BE FOUND IN TABLE.
                  *
1003 8C 1071           CPX     #TBLEND   END OF TABLE?
1006 27 04             BEQ     NOTFND    YES; BAD COMMAND RECEIVED
1008 EE 00             LDX     0,X       GOT A GOOD COMMAND. GET THE H
```

put buffer area into the accumulator for comparison with characters in the command table. The index register is used to place characters located in the command table into the B register. The two accumulators can then be compared for a match. During this process, no maskable interrupts will be processed; however, a nonmaskable interrupt will crash the program since the MPU registers will be pushed into the keyboard character buffer area since the stack pointer is pointing into that area. When this happens, the user command is destroyed, having been overwritten by the register save sequence.

We can now compare this with a new approach that still uses the stack, but uses it in its intended push-down form.

Suppose that once the command from the keyboard is received and placed into its buffer, we place a duplicate of the received command onto the stack. We do this by pushing characters in the reverse order of entry — last character first, first character last — then point the index register to the command table as was done before. If an interrupt is received, the register contents are pushed onto the stack below the keyboard characters. The characters are not destroyed by the interrupt, and the program return address is intact on the stack. This method allows fast searching yet does not alter the stack function or distort the stack operation.

Program B contains a listing of a routine using the method described. The routine performs the function of validating an input command that has been placed in the keyboard input buffer and fetching the 16-bit address of the routine that would perform the function desired. The routine is presented in a basic form and can be expanded to accommodate many useful features. One important factor to consider is that once a character

```
100A 6E 00            JMP    0,X       ADDRESS AND JUMP TO IT
                *
     100C       NOTFND EQU    *
                **
                * GO PRINT ERROR MESSAGE
                **
                *
2000                   ORG    $2000
                *
                *  PARSE SUBROUTINE
                *
     2000       PARSE  EQU    *
2000 0F                SEI              DON'T ALLOW MASKABLE INTERRUP
2001 BF 1073           STS    SAVSTK    SAVE THE STACK POINTER
2004 FF 107F           STX    SAVEX     SAVE X-REG
2007 CE 1082           LDX    #CMDTBL-1  POINT X TO COMMAND TABLE-1
                *
200A BE 1081    LOOP1  LDS    FIRST     (SP)<--ADDRESS OF FIRST KBD C
200D 34                DES
200E 08         LOOP2  INX              X POINTS TO CHAR IN CMD TABLE
200F 32                PUL A            (A)=CHAR FROM KBD
2010 E6 00             LDA B  0,X       (B)=CHAR IN CMD TABLE
2012 27 11             BEQ    GOTIT     HAVE A COMPLETE MATCH
2014 11                CBA              ARE THEY THE SAME?
2015 27 F7             BEQ    LOOP2     YES;TEST NEXT CHARS
2017 08         LOOP3  INX              NO;POINT TO NEXT IN TABLE
2018 8C 1071           CPX    #TBLEND   END OF TABLE?
201B 27 09             BEQ    BADNUZ    YES
201D E6 00             LDA B  0,X
201F 26 F6             BNE    LOOP3     SEARCH FOR 0 BYT IN TABLE
2021 08                INX              SKIP PAST UN-MATCHED HANDLER'
2022 08                INX              ADDRESS IN TABLE
2023 20 E5             BRA    LOOP1     TRY AGIAN
                *
2025 08         GOTIT  INX              POINT X TO HANDLER ADDRESS
2026 BE 1073    BADNUZ LDS    SAVSTK    RESTORE STACK POINTER
2029 01                NOP              ODD VALUE INSTR BEFORE CLI
202A 0E                CLI              ENABLE MASKABLE INTERRUPTS
202B 39                RTS              RETURN
1000                   ORG    $1000
```

*Program B. Parsing routine that allows register contents to be saved (during interrupt) below the keyboard characters.*

```
                *
                *
                ****************
                *              *
                *    PARSE2    *
                *              *
                *******************************************
                *
                *  PARSE ROUTINE WHICH PUSHES INPUT BUFFER ON STACK
                *
                *
                *
                * CALLING SEQUENCE IN MAIN PROGRAM
                *
1000 BD 1075           JSR    KEYBRD    GET CMD FROM KEYBOARD
1003 BD 100D           JSR    PARSE2
                *
                *
                * RETURN HERE; IF 0,X IS NEG, CMD NOT FOUND
                *
1006 2B 04             BMI    NOTFND
1008 08                INX              IS FOUND; GET HANDLER ADDRESS
1009 EE 00             LDX    0,X       GET TARGET ADDRESS
100B 6E 00             JMP    0,X       GO THERE
                *
                *
                * PARSE SUBROUTINE
                *
     100D       PARSE2 EQU    *
100D FE 1077           LDX    ENDBUF
1010 BF 1073           STS    SAVSTK
1013 BE 1073    PSHCHR LDS    SAVSTK
1016 A6 00             LDA A  0,X
1018 36                PSH A
```

has been pulled from the stack, it is no longer needed for comparison to a command table character set. However, if a mismatch occurs, then the same characters in the keyboard character buffer are again checked against the command table. If an interrupt occurs before a match is found, the stacked characters are lost. To avoid this situation, restack the keyboard characters each time a command table set is mismatched.

It should be clear by now that when using the stack for table manipulation and parsing, it is easy to unknowingly wipe out a program. What is needed is a parsing technique that will simulate two independent 16-bit index registers. Such a routine is described next.

Initially, set up two double-byte variables called TBLREG and KBDREG. TBLREG will be a pseudoregister in memory that will be the pointer into the lookup table. KBDREG will do the same function for the keyboard input buffer. The goal will be providing variable length table entries with a corresponding 16-bit target address. Program C is a listing of the double-index parsing routine. Several entry points are provided for accommodating various functions. Fig. 3 depicts the organization using the double-register method. This routine incorporates three loops — two inner loops and one outer loop. The pseudoregisters (KBDREG & TBLREG) must constantly be updated before fetching the other, otherwise the current register would be destroyed.

Fig. 4 details one complete entry in the command table. The method described allows entries of any length and includes associated flags and jump addresses. The process starts by initializing the pseudoregisters (FRSTCR is the address of the first nonblank character in the keyboard buffer): TABREG to the first address of the look-

```
1019 09                 DEX
101A BC 1079            CPX      FRSTCR
101D 26 F7              BNE      PSHCHR+3
101F A6 00              LDA A    0,X
1021 36                 PSH A
                 *
                 * COMPARE STACKED CHARS TO CMD TABLE
1022 CE 1082            LDX      #CMDTBL-1
1025 08          SCAN   INX
1026 32                 PUL A
1027 6D 00              TST      0,X         END OF GOOD COMMAND?
1029 27 0F              BEQ      GOTITT      YEP
102B 2B 0D              BMI      NOGOOD      BAD; END OF TABLE
102D A1 00              CMP A    0,X         ELSE; CHAR IN TABLE?
102F 27 F4              BEQ      SCAN        YES. CONTINUE
1031 08          SCAN1  INX                  NOPE. SKIP OVER UN-MATCHED TA
1032 6D 00              TST      0,X         ENTRY.  LOOK FOR 0 BYTE AS CL
1034 26 FB              BNE      SCAN1
1036 08                 INX
1037 08                 INX
1038 20 EB              BRA      SCAN
                 *
     103A        GOTITT EQU      *
                 * HAVE THE VALID COMMAND...
                 *
103A 39          NOGOOD RTS                  CAN'T FIND THE COMMAND. GET 0
```

*Program C. Double-index parsing routine.*

```
                 *
                 *
                 *******************
                 *                 *
                 *    PARSE3       *
                 *                 *
                 ********************************************
                 *
                 *
                 * DOUBLE PSEUDO REGISTER PARSE ROUTINE
                 *
     103B        PARSE3 EQU      *
103B CE 1083            LDX      #CMDTBL
103E FF 107D            STX      TABREG
1041 FE 1079     PARS4  LDX      FRSTCR
1044 FF 107B            STX      KBDREG
1047 FE 107D     PARS5  LDX      TABREG
104A E6 00              LDA B    0,X
104C 27 22              BEQ      GOTCMD
104E 2B 20              BMI      NONO
1050 08                 INX
1051 FF 107D            STX      TABREG
1054 FE 107B            LDX      KBDREG
1057 A6 00              LDA A    0,X
1059 08                 INX
105A FF 107B            STX      KBDREG
105D 11                 CBA
105E 27 E7              BEQ      PARS5
1060 FE 107D            LDX      TABREG
1063 08          PARS6  INX
1064 E6 00              LDA B    0,X
1066 26 FB              BNE      PARS6
1068 08                 INX
1069 08                 INX
106A 08                 INX
106B FF 107D            STX      TABREG
106E 20 D1              BRA      PARS4
                 *
     1070        NONO   EQU      *
                 * PRINT ERROR MESSAGE OR EXIT WITH -1 IN ACCUM
                 *
1070 39          GOTCMD RTS
                 *
                 *
                 *
1071 0002        TBLEND RMB      2
1073 0002        SAVSTK RMB      2
1075 0002        KEYBPD RMB      2
```

```
1077 0002        ENDBUF RMB    2
1079 0002        FRSTCR RMB    2
107B 0002        KBDREG RMB    2
107D 0002        TABREG RMB    2
107F 0002        SAVEX  RMB    2
1081 0002        FIRST  RMB    2
                 *
                 *
                 *   COMMAND LOOK-UP TABLE
                 *
      1083       CMDTBL EQU    *
1083 4C                 FCC    /LIST/
1084 49
1085 53
1086 54
1087 00                 FCB    0
1088 12                 FCB    $12,$34   ADDRESS FOR LIST PROCESSOR
1089 34
                 *
108A 50                 FCC    /PRI/
108B 52
108C 49
108D 00                 FCB    0
108E 56                 FCB    $56,$78   ADDRESS FOR PRINT PROCESSOR
108F 78
                 *
                        END
NOTFND 100C
PARSE  2000
LOOP1  200A
LOOP2  200E
LOOP3  2017
GOTIT  2025
BADNUZ 2026
PARSE2 100D
PSHCHR 1013
SCAN   1025
SCAN1  1031
GOTITT 103A
NOGOOD 103A
PARSE3 103B
PARS4  1041
PARS5  1047
PARS6  1063
NONO   1070
GOTCMD 1070
TBLEND 1071
SAVSTK 1073
KEYBRD 1075
ENDBUF 1077
FRSTCR 1079
KBDREG 107B
TABREG 107D
SAVEX  107F
FIRST  1081
CMDTBL 1083
```

up table and KBDREG to the address of the first non-blank character in the keyboard command buffer.

Loop 1 fetches two characters, one character into the A register from the keyboard buffer and one character into the B register from the command look-up table. The registers are tested for equality, a match between the two characters. On equality, a new pair is fetched and the process repeated until an exit condition occurs. Exit occurs if the contents of the B register is a minus one, indicating the end of the look-up table was reached before a match was made, or when the B register is zero, indicating a match is found. If no match is made in a table entry and the end of the table had not been reached, Loop 2 skips over the bytes associated with the no-match table entry. On exiting Loop 2, TABREG is updated to point to the next table entry.

This technique supports any-length table entry since the zero byte flags the end of a table entry. This method of parsing can be modified or expanded to handle most any type of command or string interpretation. Furthermore, interrupts will not alter the normal MPU interrupt process or return. ■

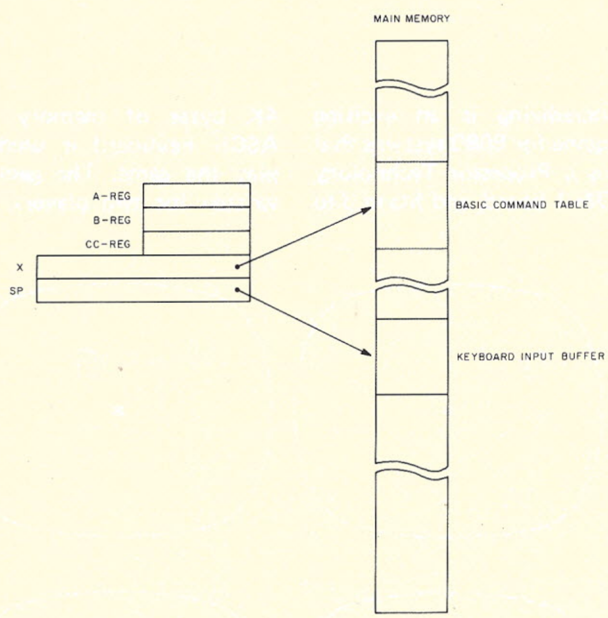



*Fig. 2. Normal parsing operation.*

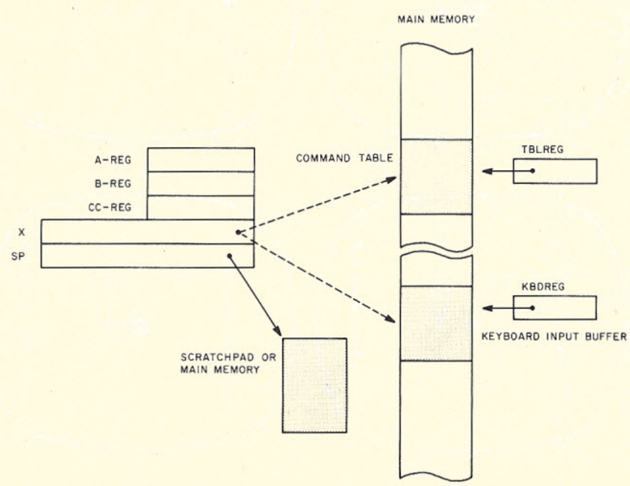



*Fig. 3. Double register parsing operation (see Program C).*

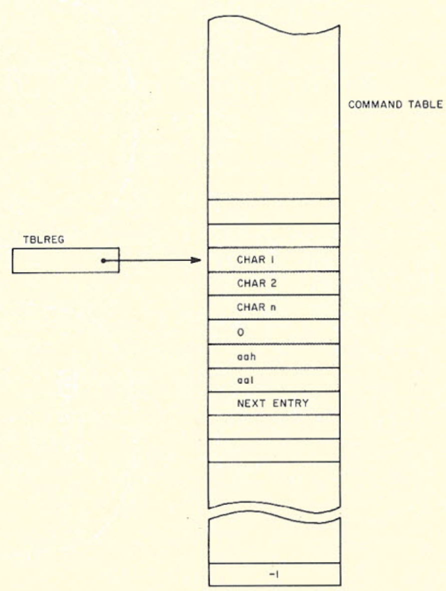



*Fig. 4. Command Table entry.*

# Incredizing

*Philip Tubb*
*ALF Products, Inc.*
*128 S. Taft*
*Lakewood CO 80228*

Incredizing is an exciting game for 8080 systems that uses a Processor Technology VDM-1 board, and fits in 3 to 4K bytes of memory. An ASCII keyboard is used to play the game. The game is written for two players, but can also be played by one. The name is, of course, a mixture of "incredible" and "amazing," and the program

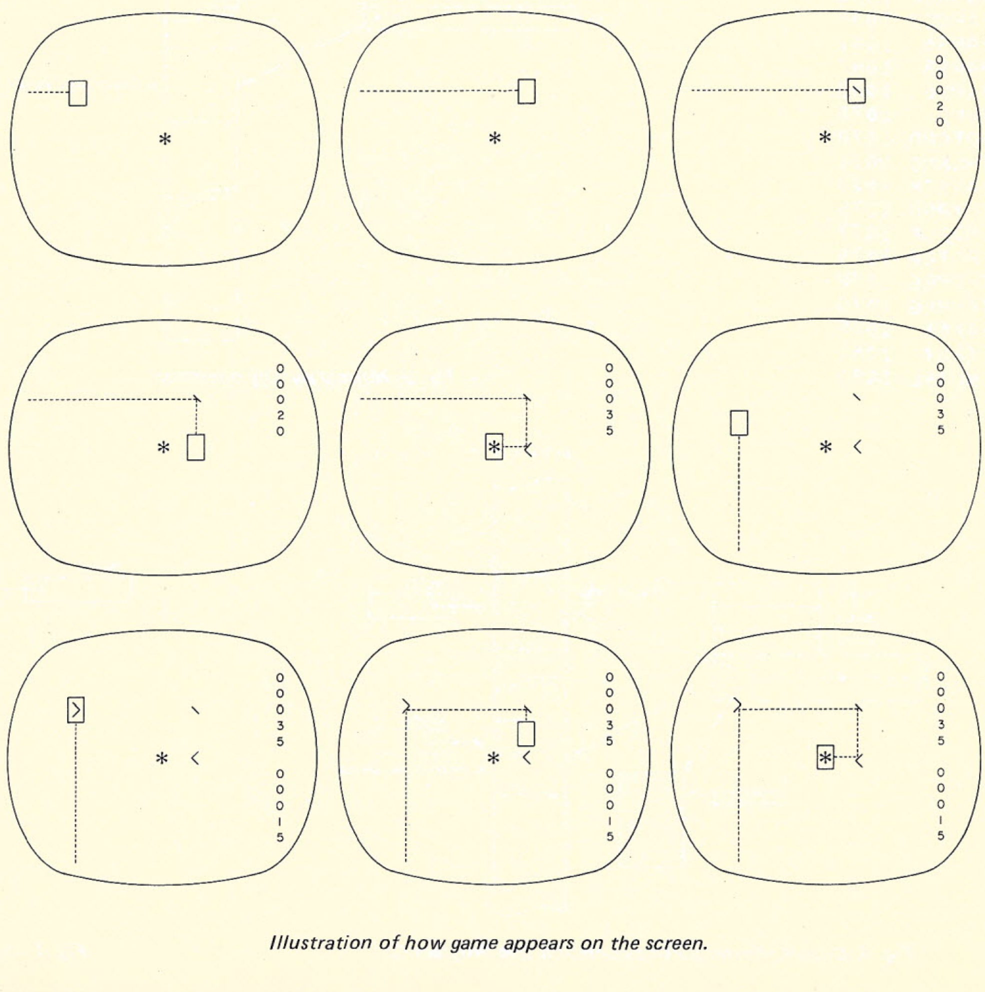



*Illustration of how game appears on the screen.*

# amazing, incredible game for 8080 systems!

is a second version of the one-player original program, *Zing*.

Here's how to play Incredizing. First, two players agree on the number of rounds to be played. The program is run (at address 020'000) and the first player presses return when he is ready to start. The screen is cleared, and a single asterisk (*) is placed on the screen. The zinger appears moving somewhere on the screen; it is a bright blob (a reverse video space).

The player tries to make the zinger hit the asterisk while scoring the least number of points possible. This is done by pressing various keys. For example, pressing a slash (/) puts a slash on the screen; and then the zinger bounces off the slash. If it was going down, it goes left; up changes to right; if it was going left it goes down; and right changes to up. There are numerous other characters used to direct the zinger.

The speed at which the zinger moves is controlled by the switch register (those without switch registers will have to arrange other inputs), and 020 octal is a reasonable

| Display | Alternate | (Approach:) Left | Up | Right | Down | (Points) On | Off |
|---|---|---|---|---|---|---|---|
| / | ? | D | R | U | L | 20 | 40 |
| \ | \| | U | L | D | R | 20 | 40 |
| - | — | UD | D | UD | U | 10 | 30 |
| ! | 1 | R | LR | L | LR | 10 | 30 |
| ' | " | UDLR | UDLR | UDLR | UDLR | 5 | 25 |
| 0 | o | hyp | hyp | hyp | hyp | 2 | 10 |
| ( | 9 | R | L | hyp | L | 4 | 8 |
| ) | 0 | hyp | R | L | R | 4 | 8 |
| < | , | R | L | UD | L | 15 | 30 |
| > | . | UD | R | L | R | 15 | 30 |
| Λ | 6 | U | D | U | LR | 15 | 30 |
| V | v | D | LR | D | U | 15 | 30 |

*Table 1. Character table.*

*Program listing.*

```
020 000                    0010 SP EQU M
020'000                    0020 PSW EQU M
020'000 061 000 000        0030 START LXI SP,0 SET STACK POINTER.
020'003 373                0040  EI ENABLE INTERRUPTS FOR CLOCK.
020'004 041 000 000        0050  LXI H,0 SET HL.
020'007 333 176            0060 START1 IN STATUS WAIT FOR CHARACTER.
020'011 043                0070  INX H COUNT WITH HL WHILE WAITING.
020'012 037                0080  RAR
020'013 322 000 020        0090  JNC START
020'016 042 163 022        0100  SHLD RND1 SET 4 BYTE RANDOM SEED.
020'021 043                0110  INX H GUARANTEE NON-0 WITH INX.
020'022 042 165 022        0120  SHLD RND2
020'025 333 177            0130  IN DATA READ CHARACTER.
020'027 346 177            0140  ANI 127
020'031 376 132            0150  CPI 'Z' IS IT Z?
020'033 312 067 020        0160  JZ BEGIN YES, LEAVE BOARD ALONE.
020'036 376 015            0162  CPI 'M'-64
020'040 302 000 020        0164  JNZ START MUST BE RETURN.
020'043                    0170 * INITIALIZE "BOARD" BY SETTING
020'043                    0180 * SCREEN TO ALL SPACES, THEN PUT
020'043                    0190 * AN "*" IN THE CENTER.
020'043 257                0200 START3 XRA A
020'044 323 310            0210  OUT 200 SET SCROLL PORT.
020'046 076 320            0220  MVI A,0D0H SET END POINT.
020'050 041 000 314        0230  LXI H,0CC00H SET START POINT.
020'053 066 040            0240 START2 MVI M,' ' ERASE.
020'055 043                0250  INX H
020'056 274                0260  CMP H
020'057 302 053 020        0270  JNZ START2 REPEAT UNTIL DONE.
020'062 076 052            0280  MVI A,'*' WRITE THE *.
020'064 062 040 316        0290  STA 0CE20H
```

```
020'067                    0300 * START OF GAME.
020'067 041 000 000        0310 BEGIN LXI H,0 ZERO SCORES FOR
020'072 042 167 022        0320  SHLD PL1 PLAYER 1,
020'075 042 171 022        0330  SHLD PL2 PLAYER2,
020'100 042 173 022        0340  SHLD PLSC CURRENT PLAYER.
020'103 041 077 315        0350  LXI H,0CD3FH SET CURRENT SCORE
020'106 042 175 022        0360  SHLD SCPT POINTER.
020'111                    0370 * BEGIN PLAY BY SETTING ZINGER AT
020'111                    0380 * A "RANDOM" PLACE. SET B TO A
020'111                    0390 * RANDOM DIRECTION.
020'111 006 010            0400 PLAY MVI B,8 SET COUNT.
020'113 315 117 022        0410 PLAY1 CALL RND GET RANDOM BIT IN CARRY.
020'116 175                0420  MOV A,L
020'117 027                0430  RAL ROTATE INTO L.
020'120 157                0440  MOV L,A
020'121 005                0450  DCR B
020'122 302 113 020        0460  JNZ PLAY1 REPEAT FOR 8 BITS.
020'125 346 077            0470  ANI 63 IN SCORE AREA?
020'127 376 077            0480  CPI 63
020'131 312 111 020        0490  JZ PLAY TRY AGAIN IF SO.
020'134 315 117 022        0500  CALL RND
020'137 076 063            0510  MVI A,51 SET A TO CC HEX / 4.
020'141 027                0520  RAL SHIFT IN RANDOM BIT.
020'142 147                0530  MOV H,A SAVE.
020'143 315 117 022        0540  CALL RND
020'146 174                0550  MOV A,H
020'147 027                0560  RAL SHIFT IN ONE MORE.
020'150 147                0570  MOV H,A COMPLETE RANDOM ADDRESS IN HL.
020'151 176                0580  MOV A,M IS IT THE ADDRESS
020'152 346 177            0590  ANI 127 OF THE *?
020'154 376 052            0600  CPI '*'
020'156 312 111 020        0610  JZ PLAY IF SO, TRY AGAIN.
020'161 315 117 022        0620  CALL RND SET 2 BIT DIRECTION IN B.
020'164 027                0630  RAL
020'165 107                0640  MOV B,A
020'166 315 117 022        0650  CALL RND
020'171 170                0660  MOV A,B
020'172 027                0670  RAL
020'173 107                0680  MOV B,A
020'174                    0690 * LIGHT UP ZINGER BY SETTING MSB (BIT
020'174                    0700 * 7) FOR REVERSE VIDEO.
020'174 176                0710 FLASH MOV A,M
020'175 366 200            0720  ORI 128
020'177 167                0730  MOV M,A
020'200 376 252            0732  CPI '*'+128
020'202 312 162 021        0734  JZ HIT BRANCH IF * HIT.
020'205                    0740 * NOW WAIT FOR A WHILE. WATCH KEYBOARD
020'205                    0750 * FOR POSSIBLE COMMANDS.
020'205                    0760 * "CLOCK" IS ADDRESS OF 2 BYTE NUMBER
020'205                    0770 * WHICH IS DECREMENTED AT REGULAR
020'205                    0780 * INTERVALS BY AN INTERRUPT ROUTINE.
020'205 333 377            0790  IN 255 READ SWITCHES.
020'207 137                0800  MOV E,A
020'210 026 000            0810  MVI D,0 USE FOR WAIT COUNTDOWN.
020'212 023                0820  INX D ELIMINATE 0 POSSIBILITY.
020'213 353                0830  XCHG
020'214 170                0840  MOV A,B READ DIRECTION.
020'215 037                0850  RAR
020'216 322 222 020        0860  JNC WAIT1
020'221 051                0870  DAD H DOUBLE WAIT IF UP OR DOWN.
020'222 042 103 000        0880 WAIT1 SHLD CLOCK SET COUNT DOWN.
020'225 333 176            0890 WAIT2 IN STATUS KEY PRESSED?
020'227 037                0900  RAR
020'230 332 270 021        0910  JC KEY JUMP IF SO.
020'233 052 103 000        0920 WAIT3 LHLD CLOCK
020'236 257                0930  XRA A
020'237 264                0940  ORA H COUNT REACHED -1 YET?
020'240 362 225 020        0950  JP WAIT2 JUMP IF NOT.
020'243                    0960 * TIME TO MOVE ZINGER. READ THE
020'243                    0970 * CHARACTER IT'S ON AND ACT
020'243                    0980 * ACCORDINGLY.
020'243 032                0990  LDAX D CLEAR MSB TO END REVERSE VIDEO.
020'244 346 177            1020  ANI 127
020'246 022                1030  STAX D
020'247 376 040            1040  CPI ' ' IS IT SPACE?
020'251 312 275 020        1050  JZ MOVE3 JUMP IF SO.
020'254 041 346 020        1060  LXI H,NORMAL
020'257 042 341 020        1070  SHLD PLACE+1
020'262 117                1080  MOV C,A
020'263 315 327 020        1090  CALL SEARCH LOOK UP CHARACTER.
020'266 267                1100  ORA A
020'267 302 064 021        1110  JNZ NEW JUMP IF FOUND.
020'272                    1120 * ILLEGAL CHARACTER ON SCREEN.
020'272 076 040            1130 MOVE2 MVI A,' ' REPLACE WITH A
020'274 022                1140  STAX D SPACE.
020'275                    1150 * SPACE, CONTINUE NORMAL MOVEMENT.
020'275 170                1160 MOVE3 MOV A,B LOAD DIRECTION.
020'276 346 003            1170  ANI 3 MASK TO 2 BITS.
020'300 207                1180  ADD A DOUBLE IT.
020'301 157                1190  MOV L,A
020'302 046 000            1200  MVI H,0
020'304 325                1210  PUSH D
```

starting speed (especially considering that is also the starting address). In addition to pressing keys to put characters on the screen players can use a space to remove them.

The amount of points scored is different for each character, as is the amount of points scored by removing the character (also, the score for putting a character on the screen is not the same as for taking it off).

Once the * is hit, it is the other player's turn. The characters put on the screen by the first player remain. After the second player hits the *, that ends the first round. After all rounds are completed, the player with the lower score wins.

According to our experiments, incredizing is played at two levels. At first, players concentrate on simply hitting the *. Later, they begin calculating not just how to hit the *, but the best way to hit it. They consider how many points each character takes and what sort of pattern they're leaving on the screen. You can set up patterns that lead to the *, so you can just move the zinger anywhere in the pattern and let it thread down to the *. You can surround the * with deflectors to make it more difficult for the other player. Very complex strategies are possible.

**Program Description**

Line 0030 in the program sets the stack pointer. We have wire-wrapped a small amount of memory at the highest addresses available to serve as a convenient stack location, and, therefore, we set the stack pointer initially to zero. You will probably have to change this to an area where you have RAM. The program is shown assembled at 020'000 (split octal, equivalent to 1000 hex) but could be assembled about anywhere.

The Processor Technology board is assumed to have the standard addresses, a starting RAM address of CC00 hex and an output port number of C8 hex. The keyboard is assumed to use the least significant bit of its status word as an input ready bit, 0 meaning not ready, and 1 meaning ready. The addresses of the status and data ports are set to 126 and 127 by EQUs at lines 3840 and 3850. The WAIT3 routine (lines 0920 through 0950) checks a two-byte word in RAM, which is supposed to be automatically decremented by an interrupt routine 256 times per second (details on this are given later). This can be replaced by a timing loop if desired. In making reverse video spaces, it is assumed that the switches on the VDM-1 are set with 2, 3, 5 and 6 on, and all others off.

If the zinger is currently at a character, and a new character is pressed, the player is charged both for removing the old character and placing the new one. The scores are shown in decimal at the far right of the screen with player one's score at the top and player two's score at the bottom. The score is not shown when it is zero.

At the start of the game, Incredizing waits for either of two input characters. Return initializes the screen and begins play. Capital Z starts the game without initializing the screen. Each time the * is hit, the new player indicates he is ready by pressing return. If he presses capital Z, the screen is cleared and the game starts over.

The random number generator generates a single-bit number. This is compatible with practically any hardware random number generator, and the random numbers actually used are made by calling the routine as many times as needed.

The character table, which begins at line 6000, is easily modified to create your own special characters, accommodate different keyboards or modify existing special

```
020'305 021 317 020     1220  LXI D,BRANCH ADD BRANCH TABLE ADDRESS.
020'310 031             1230  DAD D
020'311 136             1240  MOV E,M READ ADDRESS FROM TABLE.
020'312 043             1250  INX H
020'313 126             1260  MOV D,M
020'314 353             1270  XCHG
020'315 321             1280  POP D
020'316 351             1290  PCHL BRANCH.
020'317                 1300 * BRANCH TABLE FOR ABOVE ROUTINE.
020'317 360 020         1310 BRANCH DW LEFT
020'321 013 021         1320  DW UP
020'323 375 020         1330  DW RIGHT
020'325 040 021         1340  DW DOWN
020'327                 1350 * ROUTINE TO SEARCH CHARACTER TABLE. LXI H,NORMAL FOR
020'327                 1360 * NORMAL SEARCH OR LXI H,BOTH FOR INPUT SEARCH; THEN
020'327                 1370 * SHLD PLACE+1. PUT CHARACTER TO BE FOUND IN C.
020'327                 1380 * ROUTINE RETURNS A AS 0 IF NOT FOUND, ELSEWISE IT
020'327                 1390 * SETS HL TO POINT TO 2ND BYTE OF ENTRY.
020'327 041 177 022     1400 SEARCH LXI H,TABLE SET POINTER.
020'332 176             1410 SEAR1 MOV A,M
020'333 267             1420  ORA A
020'334 310             1430  RZ RETURN ON END MARKER.
020'335 271             1440  CMP C CHARACTER FOUND?
020'336 043             1450  INX H
020'337 310             1460  RZ RETURN IF SO.
020'340 303 346 020     1470 PLACE JMP NORMAL (SOMETIMES IS JMP BOTH)
020'343 176             1480 BOTH MOV A,M
020'344 271             1490  CMP C CHECK INPUT CHARACTER.
020'345 310             1500  RZ RETURN ON MATCH.
020'346 043             1510 NORMAL INX H
020'347 043             1520  INX H
020'350 043             1530  INX H
020'351 043             1540  INX H
020'352 043             1550  INX H POINT TO NEXT ENTRY.
020'353 303 332 020     1560  JMP SEAR1 CONTINUE SEARCH.
020'356 006 000         1565 RIGHTX MVI B,0 CHANGE DIRECTION.
020'360                 1570 * MOVE LEFT ROUTINE.
020'360 173             1580 LEFT MOV A,E AT LEFT EDGE?
020'361 346 077         1590  ANI 63
020'363 312 373 020     1600  JZ LEFTX CHANGE TO RIGHT IF SO.
020'366 033             1610  DCX D MOVE LEFT.
020'367 353             1620 CONT XCHG PUT NEW ADDRESS IN HL.
020'370 303 174 020     1630  JMP FLASH CONTINUE.
020'373 006 002         1640 LEFTX MVI B,2 CHANGE DIRECTION.
020'375                 1650 * MOVE RIGHT ROUTINE
020'375 173             1660 RIGHT MOV A,E AT RIGHT EDGE
020'376 346 077         1670  ANI 63 (EXCLUDING SCORE AREA)?
021'000 376 076         1680  CPI 62
021'002 312 356 020     1690  JZ RIGHTX CHANGE TO LEFT IF SO.
021'005 023             1700  INX D MOVE RIGHT.
021'006 303 367 020     1710  JMP CONT CONTINUE.
021'011 006 001         1720 DOWNX MVI B,1 CHANGE DIRECTION.
021'013                 1730 * MOVE UP ROUTINE.
021'013 172             1740 UP MOV A,D AT TOP?
021'014 376 314         1750  CPI 0CCH
021'016 312 030 021     1760  JZ UP2 JUMP IF MAYBE.
021'021 041 300 377     1770 UP1 LXI H,-64 MOVE UP.
021'024 031             1780 CONT1 DAD D
021'025 303 174 020     1790  JMP FLASH CONTINUE.
021'030 173             1800 UP2 MOV A,E AT TOP?
021'031 346 300         1810  ANI 192
021'033 302 021 021     1820  JNZ UP1 JUMP IF NOT.
021'036 006 003         1830  MVI B,3 CHANGE DIRECTION.
021'040                 1840 * MOVE DOWN ROUTINE.
021'040 172             1850 DOWN MOV A,D AT BOTTOM?
021'041 376 317         1870  CPI 0CFH
021'043 302 056 021     1880  JNZ DOWN1 JUMP IF NOT.
021'046 173             1890  MOV A,E AT BOTTOM?
021'047 346 300         1900  ANI 192
021'051 376 300         1910  CPI 192
021'053 312 011 021     1920  JZ DOWNX JUMP IF SO.
021'056 041 100 000     1930 DOWN1 LXI H,64 MOVE DOWN.
021'061 303 024 021     1940  JMP CONT1 CONTINUE.
021'064                 1950 * PROCESS NEW DIRECTION CHARACTERS.
021'064 043             1960 NEW INX H POINT TO FIRST DIRECTION.
021'065 043             1970  INX H
021'066 043             1980  INX H
021'067 170             1990  MOV A,B LOAD CURRENT DIRECTION.
021'070 037             2000  RAR
021'071 037             2010  RAR
021'072 322 076 021     2020  JNC NEW1 SKIP BYTE IF RIGHT OR DOWN.
021'075 043             2030  INX H
021'076 027             2040 NEW1 RAL
021'077 176             2050  MOV A,M
021'100 332 107 021     2060  JC NEW2
021'103 037             2070  RAR USE LEFT HALF IF LEFT OR RIGHT.
021'104 037             2080  RAR
021'105 037             2090  RAR
021'106 037             2100  RAR
021'107 346 017         2110 NEW2 ANI 15 MASK TO 4 BITS.
021'111                 2120 * BITS 3 THROUGH 0 NOW CONTAIN
021'111                 2130 * "ACCEPTABLE NEW DIRECTION" BITS:
021'111                 2140 * BIT 3 FOR DOWN, 2 FOR RIGHT, 1
```

```
021'111                  2150 * FOR UP, AND BIT 0 FOR LEFT. IF A
021'111                  2160 * BIT IS 1, THAT DIRECTION IS ALLOWED,
021'111                  2170 * IF 0 IT IS NOT. ALL 0'S (NO ALLOWED
021'111                  2180 * DIRECTIONS) INDICATES A HYPERSPACE
021'111                  2190 * JUMP.
021'111 312 111 020      2200  JZ PLAY JUMP IF HYPERSPACE.
021'114 117              2210  MOV C,A SAVE IN C.
021'115 315 117 022      2220  CALL RND PUT 2 BIT RANDOM NUMBER
021'120 027              2230  RAL IN H.
021'121 147              2240  MOV H,A
021'122 315 117 022      2250  CALL RND
021'125 174              2260  MOV A,H
021'126 027              2270  RAL
021'127 346 003          2280  ANI 3
021'131 147              2290  MOV H,A
021'132 044              2292  INR H
021'133 171              2300  MOV A,C PREPARE FOR SHIFTS.
021'134 006 377          2310  MVI B,255
021'136 267              2312  ORA A CLEAR CARRY.
021'137 322 144 021      2320 NEW3 JNC NEWX WRAP CARRY INTO BIT 3.
021'142 366 010          2322  ORI 8
021'144 037              2324 NEWX RAR
021'145 004              2330  INR B KEEP TRACK OF DIRECTION.
021'146 045              2340  DCR H
021'147 302 137 021      2350  JNZ NEW3 DO A RANDOM # OF TIMES.
021'152 332 275 020      2360 NEW4 JC MOVE3 GO MOVE IF NEW
021'155 004              2370  INR B DIRECTION IS LEGAL, ELSE
021'156 037              2380  RAR TRY NEXT BIT.
021'157 303 152 021      2390  JMP NEW4
021'162                  2400 * TARGET HIT. WAIT FOR KEY PRESSED AND
021'162                  2410 * SWITCH PLAYERS.
021'162 333 177          2420 HIT IN DATA EAT POSSIBLE PREVIOUS CHAR.
021'164 353              2422  XCHG ADDRESS OF * IN DE.
021'165 072 176 022      2430  LDA SCPT+1 LOAD SCORE POINTER.
021'170 376 315          2440  CPI OCDH 1ST PLAYER?
021'172 052 173 022      2450  LHLD PLSC
021'175 312 251 021      2460  JZ HIT4 JUMP IF SO.
021'200 042 171 022      2470  SHLD PL2 2ND, STORE SCORE.
021'203 052 167 022      2480  LHLD PL1 STORE OTHER PLAYER'S.
021'206 042 173 022      2490  SHLD PLSC
021'211 041 077 315      2500  LXI H,OCD3FH SET NEW SCORE POINTER.
021'214 042 175 022      2510 HIT2 SHLD SCPT
021'217 333 176          2520 HIT3 IN STATUS WAIT FOR INPUT.
021'221 037              2530  RAR
021'222 322 217 021      2540  JNC HIT3
021'225 333 177          2570  IN DATA
021'227 346 177          2580  ANI 127
021'231 376 132          2590  CPI 'Z' A Z?
021'233 312 043 020      2600  JZ START3 IF SO, NEW GAME.
021'236 376 015          2602  CPI 'M'-64
021'240 302 217 021      2604  JNZ HIT3 (MUST BE RETURN.)
021'243 076 052          2606  MVI A,'*' RESET TARGET.
021'245 022              2608  STAX D
021'246 303 111 020      2610  JMP PLAY ELSE CONTINUE.
021'251 042 167 022      2620 HIT4 SHLD PL1 STORE SCORE.
021'254 052 171 022      2630  LHLD PL2 STORE OTHER PLAYER'S.
021'257 042 173 022      2640  SHLD PLSC
021'262 041 377 317      2650  LXI H,OCFFFH SET SCORE POINTER.
021'265 303 214 021      2660  JMP HIT2 CONTINUE.
021'270                  2670 * PROCESS KEY PRESSED.
021'270 333 177          2680 KEY IN DATA READ KEY.
021'272 346 177          2690  ANI 127 KILL PARITY.
021'274 041 301 021      2692  LXI H,KEY3+2 (POINT TO FAKE SPACE.)
021'277 376 040          2694 KEY3 CPI ' '
021'301 312 322 021      2696  JZ KEY2 JUMP IF SPACE.
021'304 117              2700  MOV C,A PUT IN C FOR SEARCH.
021'305 041 343 020      2710  LXI H,BOTH
021'310 042 341 020      2720  SHLD PLACE+1
021'313 315 327 020      2730  CALL SEARCH LOOK FOR CHARACTER.
021'316 267              2740  ORA A
021'317 312 233 020      2750  JZ WAIT3 JUMP IF NOT FOUND.
021'322 345              2760 KEY2 PUSH H SAVE POINTER.
021'323 032              2770  LDAX D LOAD DISPLAY CHARACTER.
021'324 346 177          2775  ANI 127
021'326 117              2780  MOV C,A PUT IN C FOR SEARCH.
021'327 041 346 020      2790  LXI H,NORMAL
021'332 042 341 020      2800  SHLD PLACE+1
021'335 315 327 020      2810  CALL SEARCH LOOK FOR IT.
021'340 267              2820  ORA A
021'341 312 362 021      2830  JZ KEY1 JUMP IF NOT FOUND.
021'344                  2840 * VALID KEY PRESSED AND VALID CHAR.
021'344                  2850 * IS ON SCREEN. REMOVE SCREEN CHAR.
021'344                  2860 * AND RACK UP THE POINTS.
021'344 043              2870  INX H
021'345 043              2880  INX H POINT TO POINTS.
021'346 305              2885  PUSH B
021'347 116              2890  MOV C,M
021'350 006 000          2900  MVI B,0
021'352 052 173 022      2910  LHLD PLSC
021'355 011              2920  DAD B
021'356 301              2925  POP B
021'357 042 173 022      2930  SHLD PLSC
021'362                  2940 * PROCESS VALID KEY.
```

characters. The table is arranged for the IBM keyboard rather than the ANSI standard keyboard because we are using a Dec-writer II for input. The character table is described below.

The interrupt routine, which begins at line 9000, simply decrements the two-byte number stored at CLOCK, which follows the routine. It is assumed that RST 7 will be used to call the routine.

### The Character Table

The character table has one entry for each special character. It ends with an end marker, which consists of a single zero. Each special character is defined and the possible movements for it are specified. For each approach direction (up, down, left, and right) the possible new directions are specified. Any combination of up, down, left and/or right may be specified, and the new direction will be picked at random from the possible legal directions. If no new directions are allowed, then the zinger goes into hyperspace; it emerges at a random place on the screen moving in a random direction. Each character entry is as follows:

1. The first byte indicates the character that will be displayed on the screen. It is also one of two acceptable input characters. That is, if either it or the other acceptable input character is typed in, the character defined in the first byte will be displayed.

2. The second byte indicates the other acceptable input character. This is normally the display character either shifted or not shifted. This allows the player to ignore the shift key when playing. Both characters should have parity zero (decimal values less than 128).

3. The third byte indicates the number of points scored when a player puts the character on the screen.

4. The fourth byte indi-

cates the number of points scored when a player takes the character off the screen.

5. The fifth byte indicates the new movement when the approach direction was either left or up.

6. The sixth byte indicates the new movement when the approach direction was either right or down.

For the fifth and sixth bytes, the most significant four bits of the entry are used for left or right approach directions, and the least significant four are for approach directions of up or down. For each four-bit part, a bit should be 1 to allow the direction, or 0 to disallow it, and the directions are (from most significant bit to least significant): down, right, up and left.

**Current Zing Characters**

In Table 1, the first character is the character displayed on the screen. The first and second characters are acceptable inputs to display the first character. Under the left, up, right and down columns, the possible new directions that may occur when the character is hit while going in the approach direction are given as U for up, D for down, L for left, R for right and hyp for hyperspace. Under the on and off columns, the point values are given.

**Interrupts and Timing**

Fig. 1 shows the circuit for a timer that can easily be wire-wrapped for use with the Altair bus. It consists of an NE555 timer chip, a 7474 or 74LS74 flip-flop (one half of which is used to provide a complimentary output), two fixed resistors, two capacitors and one variable resistor (trimmer). The output of pin 3 of the NE555 is adjusted to approximately 256 hertz by adjusting the variable resistor the actual frequency is not very important as it will only affect the speed of the zinger.

At each clock pulse from the NE555, the $\overline{\text{PINT}}$ line of

```
021'362 341          KEY1 POP H RECOVER TABLE POINTER.
021'363 053           DCX H POINT TO DISPLAY CHARACTER.
021'364 176           MOV A,M LOAD IT.
021'365 366 200       ORI 128 SET ZINGER.
021'367 022           STAX D PUT ON SCREEN.
021'370 376 240       CPI ' '+128
021'372 312 013 022   JZ KEY4 JUMP IF SPACE.
021'375 043           INX H
021'376 043           INX H
021'377 305           PUSH B
022'000 116           MOV C,M LOAD SCORE.
022'001 006 000       MVI B,0
022'003 052 173 022   LHLD PLSC
022'006 011           DAD B ADD CORRECT AMOUNT.
022'007 301           POP B
022'010 042 173 022   SHLD PLSC
022'013              * UPDATE SCORE ON SCREEN.
022'013 325          KEY4 PUSH D
022'014 305           PUSH B
022'015 052 175 022   LHLD SCPT LOAD SCORE POINTER.
022'020 345           PUSH H SAVE.
022'021 036 005       MVI E,5 SET COUNTER.
022'023 052 173 022   LHLD PLSC LOAD SCORE.
022'026 016 021      SCORE1 MVI C,17
022'030 026 000       MVI D,0
022'032 172           MOV A,D
022'033 326 012      DIV1 SUI 10
022'035 322 042 022   JNC DIV2
022'040 306 012       ADI 10
022'042 077          DIV2 CMC
022'043 127           MOV D,A
022'044 170           MOV A,B
022'045 027           RAL
022'046 107           MOV B,A
022'047 175           MOV A,L
022'050 027           RAL
022'051 157           MOV L,A
022'052 174           MOV A,H
022'053 027           RAL
022'054 147           MOV H,A
022'055 172           MOV A,D
022'056 027           RAL
022'057 015           DCR C
022'060 302 033 022   JNZ DIV1
022'063 037           RAR
022'064 145           MOV H,L
022'065 150           MOV L,B
022'066 042 341 020   SHLD PLACE+1 SAVE HL/10.
022'071 341           POP H RECOVER SCORE POINTER.
022'072 306 060       ADI '0' PUT REMAINDER IN ASCII.
022'074 167           MOV M,A PUT ON SCREEN.
022'075 001 300 377   LXI B,-64
022'100 011           DAD B POINT TO NEXT PLACE.
022'101 345           PUSH H SAVE H AGAIN.
022'102 052 341 020   LHLD PLACE+1 RECOVER HL/10.
022'105 035           DCR E
022'106 302 026 022   JNZ SCORE1 REPEAT 5 TIMES.
022'111 341           POP H
022'112 301           POP B
022'113 321           POP D GET EVERYTHING BACK.
022'114 303 233 020   JMP WAIT3 CONTINUE.
022'117              * RANDOM BIT GENERATOR ROUTINE. PUTS ONE BIT
022'117              * INTO CARRY. (THIS ALLOWS A SIMPLE 1-BIT
022'117              * HARDWARE RANDOM TO BE USED INSTEAD.)
022'117 345          RND PUSH H
022'120 325           PUSH D
022'121 052 163 022   LHLD RND1
022'124 353           XCHG
022'125 052 165 022   LHLD RND2
022'130 172          RNDA MOV A,D
022'131 037           RAR
022'132 255           XRA L
022'133 037           RAR
022'134 037           RAR
022'135 172           MOV A,D
022'136 027           RAL
022'137 127           MOV D,A
022'140 173           MOV A,E
022'141 037           RAR
022'142 137           MOV E,A
022'143 174           MOV A,H
022'144 037           RAR
022'145 147           MOV H,A
022'146 175           MOV A,L
022'147 037           RAR
022'150 157           MOV L,A
022'151 042 165 022   SHLD RND2
022'154 353           XCHG
022'155 042 163 022   SHLD RND1
022'160 321           POP D
022'161 341           POP H
022'162 311           RET
022'163              * EQU'S
```

```
022'163              3840 STATUS EQU 126
022'163              3850 DATA EQU 127
022'163              3870 * TEMPORARIES
022'163              3880 RND1 DS 2
022'165              3890 RND2 DS 2
022'167              3900 PL1 DS 2
022'171              3910 PL2 DS 2
022'173              3920 PLSC DS 2
022'175              3930 SCPT DS 2
022'177              6000 * CHARACTER TABLE.
022'177 057          6100 TABLE DB '/'
022'200 077          6110  DB '?'
022'201 024          6120  DB 20
022'202 050          6130  DB 40
022'203 204          6140  DB 84H
022'204 041          6150  DB 21H
022'205 134          6160  DB 'Z'+2 (BACK SLASH)
022'206 174          6170  DB 'Z'+34 (VERTICAL BAR)
022'207 024          6180  DB 20
022'210 050          6190  DB 40
022'211 041          6200  DB 21H
022'212 204          6210  DB 84H
022'213 055          6220  DB '-'
022'214 137          6230  DB 95 (UNDERLINE)
022'215 012          6240  DB 10
022'216 036          6250  DB 30
022'217 250          6260  DB 0A8H
022'220 242          6270  DB 0A2H
022'221 041          6280  DB '!'
022'222 061          6290  DB '1'
022'223 012          6300  DB 10
022'224 036          6310  DB 30
022'225 105          6320  DB 45H
022'226 025          6330  DB 15H
022'227 047          6340  DB 39 (')
022'230 042          6350  DB '"'
022'231 005          6360  DB 5
022'232 031          6370  DB 25
022'233 377          6380  DB 0FFH
022'234 377          6390  DB 0FFH
022'235 117          6400  DB 'O'
022'236 157          6410  DB 'O'+32
022'237 002          6420  DB 2
022'240 012          6430  DB 10
022'241 000          6440  DB 0H
022'242 000          6450  DB 0H
022'243 050          6460  DB '('
022'244 071          6470  DB '9'
022'245 004          6480  DB 4
022'246 010          6490  DB 8
022'247 101          6500  DB 41H
022'250 001          6510  DB 1H
022'251 051          6520  DB ')'
022'252 060          6530  DB '0'
022'253 004          6540  DB 4
022'254 010          6550  DB 8
022'255 004          6560  DB 4H
022'256 024          6570  DB 14H
022'257 074          6580  DB '<'
022'260 054          6590  DB ','
022'261 017          6600  DB 15
022'262 036          6610  DB 30
022'263 101          6620  DB 41H
022'264 241          6630  DB 0A1H
022'265 076          6640  DB '>'
022'266 056          6650  DB '.'
022'267 017          6660  DB 15
022'270 036          6670  DB 30
022'271 244          6680  DB 0A4H
022'272 024          6690  DB 14H
022'273 136          6700  DB 94 (UP ARROW)
022'274 066          6710  DB '6'
022'275 017          6720  DB 15
022'276 036          6730  DB 30
022'277 050          6740  DB 28H
022'300 045          6750  DB 25H
022'301 126          6760  DB 'V'
022'302 166          6770  DB 'V'+32
022'303 017          6780  DB 15
022'304 036          6790  DB 30
022'305 205          6800  DB 85H
022'306 202          6810  DB 82H
022'307 000          6820  DB 0 END MARKER.
022'310              9000 * INTERRUPT ROUTINE AT OCTAL 70
022'310              9010  ORG 56
000'070 345          9020  PUSH H
000'071 052 103 000  9030  LHLD CLOCK
000'074 053          9040  DCX H
000'075 042 103 000  9050  SHLD CLOCK
000'100 341          9060  POP H
000'101 373          9070  EI
000'102 311          9080  RET
000'103              9090 CLOCK DS 2
```

the Altair bus is pulled low to generate an interrupt. It is held low until the SINTA signal (interrupt acknowledge) occurs. This circuit must be used only if a priority interrupt chip is not used, and no other interrupting devices exist. If a priority interrupt card is used, you may be able to connect the output of the flip-flop (pin 6) to interrupt line 7 to allow the interrupt card to work, but no other devices may use line 7. As mentioned in the text above, this circuit may be replaced by a timing loop in place of the interrupt routine; but using the circuit ensures more accurate movement of the zinger.

Paper tapes of the source and object of Incredizing are available for a $5 reproduction and postage charge from: ALF Products, Inc., Attn.: Philip Tubb, 128 South Taft, Denver CO 80228. ■

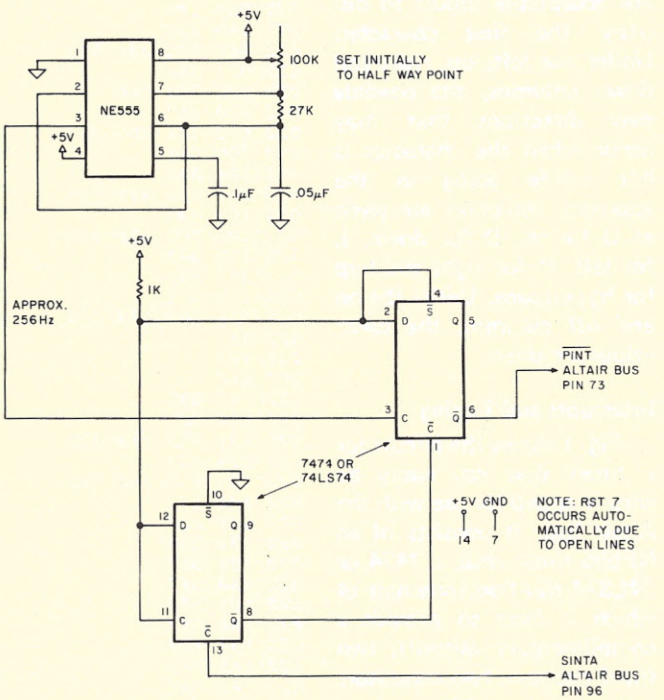
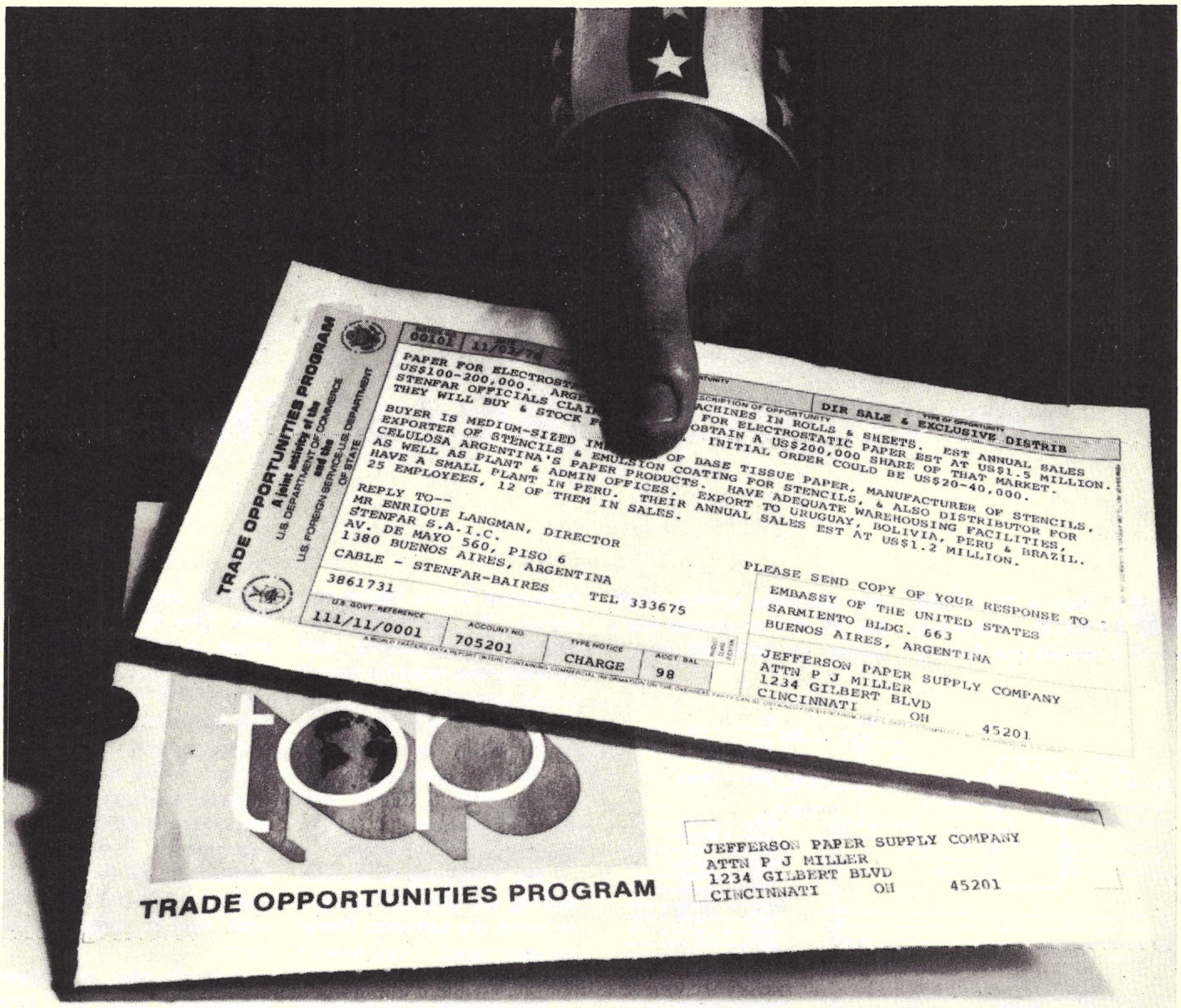



*Fig. 1. Timer circuit.*




# To get overseas trade leads like this, you could open offices in 127 countries.

# Or use our computer.

The U.S. Commerce Department's computer-operated Trade Opportunities Program (TOP) can supply you with immediate, continuing, specific leads tailored to your sales objectives for any of 127 countries. So if you can't be all over the world at one time, we've got the answer. Send us the coupon now and see how you can find where the live leads are living.

Secretary of Commerce
U.S. Department of Commerce, BIC-9A
Washington, D.C. 20230

Please tell me more about the Trade Opportunities Program (TOP).

Name ____________________

Title ____________________

Company ____________________

Address ____________________

City __________ State __________ Zip __________

*Harley D. Johnson*
*20415 S. Patsy Dr.*
*Oregon City OR 97045*

# Avoid Program Loading and Reloading

## *new 4K EPROM board from SWTP*

**W**hy buy a 4K 1702A EPROM memory board? I needed one for my new operating system, which has all the most-used entry points of MIKBUG plus a formatted binary loader and dump for cassette tape, EPROM programming and verifying and word-formatted dump-to-printer routine. These are some of the possibilities for firmware that can be used in any system. Or, how about a floppy disk operating system? . . . Let your imagination run wild.

The first thing to do after buying the EPROM board is to write the software. Sharpen your pencil if you haven't already.

The second is to program the 1702s. There are various options, including having a friend do it for you. If your friend has no programmer, have Morrow, Godbout, Cramer, Almac/Stroum or other microcomputer distributor in your area do it for you. The price is nominal. I paid, at one time, about $5 to have one 1702 programmed.

The SWTPC 6800 has needed some new kits and boards to complement the existing system for a long time. One of the newest to arrive is the APTEC 4K 1702A EPROM memory board. This board will provide added resident firmware to your system.

The board is of quality design, featuring an unbroken ground plane surrounding the top side of a double-sided plated-through board. The ground plane runs between each row of integrated circuits.

Liberal use of bypass capacitors and the good ground plane minimize noise, and the board will accept up to 16 1702A EPROMs for a total of 4096 bytes.

Access time of the 1702 EPROMs may be too slow for the 1 MHz system clock. This possible problem may be solved by using the Ø2 clock stretcher as described by Jerry Henshaw of APTEC in the December 1976 issue of *Byte*.

The slow memory line

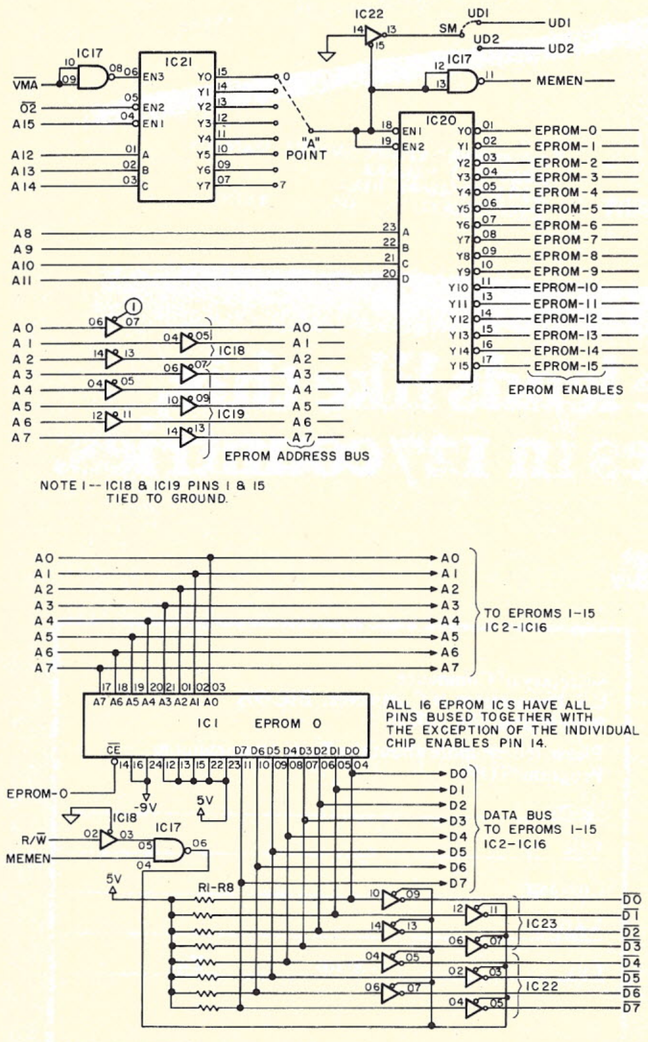



*Fig. 1. APTEC 4K EPROM board schematic. The board can be strapped to reside in any 4K boundary in the first 32K of memory via jumper at point A.*

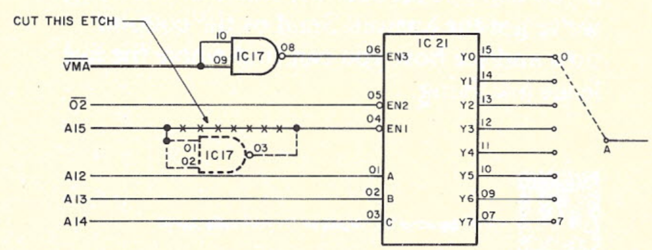
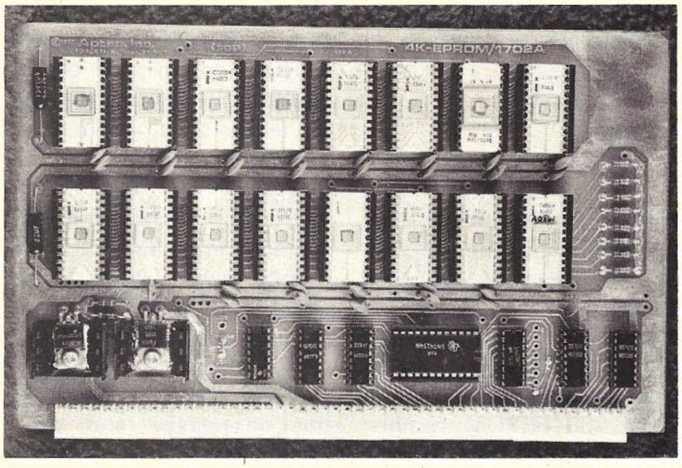



*Fig. 2. Modification to allow board to reside in upper 32K of memory. Involves cutting of etch and wiring in unused gate of IC17.*



*APTEC EPROM memory completed, EPROMs installed and ready to run.*

| IC No. | EPROM No. | Address Range |
|---|---|---|
| 1 | 0 | *X000 - X0FF |
| 2 | 1 | X100 - X1FF |
| 3 | 2 | X200 - X2FF |
| 4 | 3 | X300 - X3FF |
| 5 | 4 | X400 - X4FF |
| 6 | 5 | X500 - X5FF |
| 7 | 6 | X600 - X6FF |
| 8 | 7 | X700 - X7FF |
| 9 | 8 | X800 - X8FF |
| 10 | 9 | X900 - X9FF |
| 11 | 10 | XA00 - XAFF |
| 12 | 11 | XB00 - XBFF |
| 13 | 12 | XC00 - XCFF |
| 14 | 13 | XD00 - XDFF |
| 15 | 14 | XE00 - XEFF |
| 16 | 15 | XF00 - XFFF |

* The X is the strapping at point A on the board 0-7; a strap at 3, for example, sets the board for 3000-3FFF. With the circuit change for the upper 32K of memory, the strapping would equal 8-F.

*Fig. 3. EPROM Address Map.*

from the Ø2 clock stretcher board is brought in via the UD1 or UD2 (user-defined) bus on the motherboard to the EPROM board.

The board can be strapped to reside in any 4K boundary in the first 32K of addressable memory (see Fig. 1). The upper 32K of memory can be accessed by a simple circuit modification (see Fig. 2). The address map in Fig. 3 will be helpful.

### How It Works

As seen in Fig. 1, address decoding is done by IC20 and IC21. IC21 is a three-to-eight-line decoder, which is enabled by ANDing the Valid Memory Address (VMA), Ø2 clock and the complement of address line A15; also decoded are address lines A12-A14. Any of the outputs of IC21 can be strapped to point A on the board to provide the eight 4K boundaries (see Fig. 3). This point A is used to enable IC20, a 4-to-16-line decoder and to signal the Ø2 clock stretcher to slow the clock (SM). IC20 decodes address lines A8-A11 for addressing one of the 16 EPROM locations.

Address lines A0 - A7 are buffered by IC18 and IC19; these form the address lines for the EPROMs.

The data buses are Tri-state buffered out of the EPROMs by IC22 and IC23, and are enabled by a memory-read cycle from IC17. All data lines from the EPROMs are pulled up to +5 V through 1k resistors R1 - R8.

### Programming Formats

There are many EPROM programming formats commonly in use. The most popular are Mark Sense cards and BPNF, using punched paper tape.

The Mark Sense card, the size of an IBM-style punch card, has 32 fields of eight bits per field (see Fig. 4).

To develop a truth table for programming the 1702, you need to fill in small rectangles in each field. A soft lead pencil is used; no. 1 or 2 hardness is suitable. A filled-in bit equals a logical one (1); no mark equals a logical zero (0). If an error is made, erase the mark well or it may be sensed as a one (1). If many corrections are made, start over with a new card to ensure a good program.

BPNF format uses punched paper tape to identify the bit pattern to be programmed. The character format rules are as follows: B start character, F stop character, P data bit logical one (1) and N data bit logical zero (0). A typical punch format is shown in Fig. 5. The format requires the following:

1. Exactly 256 word fields in consecutive sequence, starting with word field 0 and ending with word field 255. If only a portion of the EPROM is to be programmed, the same format requirements apply.
2. A word field must contain ten of the format characters, with eight data characters framed with a start B and a stop F. If you make an error and haven't typed an F, type a B and retype the eight data characters followed by F.

If any character other than P or N is typed, it is an error and should be typed over with rubouts.

3. A leader and a trailer of at least 25 rubouts precede the first word field and follow the last word field.
4. A carriage return and line feed need to be inserted before each word field or at least between every four

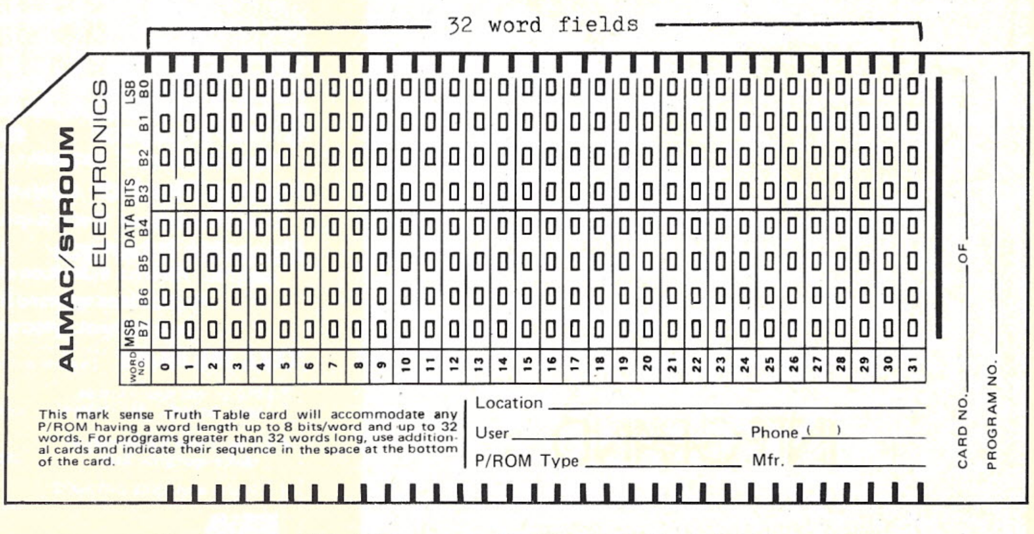



*Fig. 4. A typical Mark Sense card with 32 word fields of eight bits per field.*

```
Start Character   Stop Character      Data Field        MSB          LSB
         |             |              |        |          |            |
         BPNNNPPNPF ............. BPPPPNNNPF ...... BPNPNPNPNF
```

*Fig. 5. BPNF punch-tape format showing typical word field.*

word fields. This is to help in error checking. A word number as a "comment" at every word field or every four word fields is desirable. A comment may not contain Bs or Fs. See Fig. 6, format checking.

If you are serious about programming EPROMs for yourself or others, I suggest writing the software; and let your computer do it for you.

Computer controlled: Szerlip Enterprises, 1414 W. 259th St., Harbor City CA 90710. Kit price $165.

Manually controlled: Associated Electronics, 12444 Lambert Circle, Garden Grove CA 92614. Kit price $189.95.

*Table 1. EPROM programmers.*

There are various other 1702 programmers on the market. Table 1 lists two. The first is computer controlled. The second is a manually operated keyboard-type with hex display.

**Conclusion**

The APTEC EPROM board is straightforward in design and relatively easy to assemble (see Photo). Then, an hour or so to load your EPROMs with your software . . . and you're ready to go.

The 4K EPROM is available as a kit, with all ICs, (less 1702A), sockets and edge connectors, resistors and capacitors, for $87.50, or board only and edge connector for $27.50. The Ø2 clock stretcher in kit or board only is available for $6.25 and $2.50, respectively. Kits and boards are shipped postpaid in about two to four weeks.

Another new product from APTEC for the SWTPC 6800 is in the works. It's a 2704/2708 EPROM memory board, with at least 8K and on-board programming facilities. A programming subroutine, or possibly a bootstrap EPROM option, will be included. This new board is due for release soon, according to APTEC.

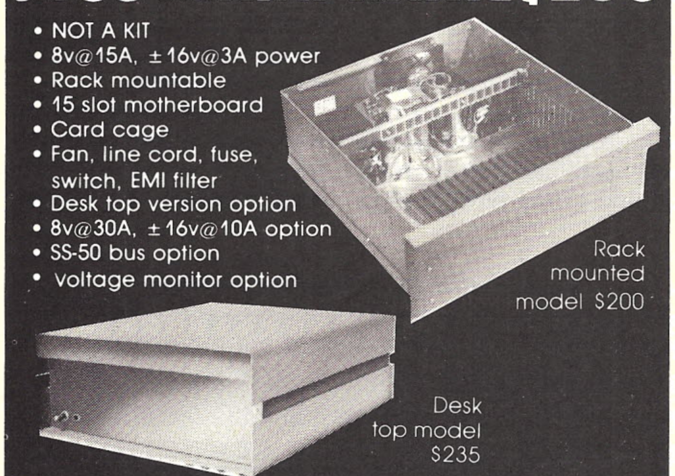
After using the 1702A EPROM board and not having a programmer available when I need one, I know I can easily use a 2704/2708 board with on-board programming.

(Thanks to Jerry Henshaw, APTEC, Inc., for providing the photograph and schematics of the 4K EPROM board.) ■

```
                                           Number "comment"
0000    BPPPNNNPPF
0001    BNNNNNNPPF
0002    BPPPPPPPPF          Word field 0 through 255.
0003    BNNPPPPPPF          CR/LF before each word field.
..................
0255    BPPNNNPPPF
```

*Fig. 6. Typed out format checking done by reading punched tape back to TTY.*










*James McClure*
*1019 Van Kirk St.*
*Philadelphia PA 19149*

# Time-sharing for the Home System

## *running two programs at once*

```
                                        ; Title — LINK
                                        ; by Jim McClure
                                        ;
                                        ; July 1977
                                        ;
                                        ;
            ORG      XXXXH              ;starting address may be anywhere
STKA        EQU      ____ H             ;initial stack pointer for PRGMA
STKB        EQU      ____ H             ;initial stack pointer for PRGMB
PRGMA       EQU      ____ H             ;start address of first program
PRGMB       EQU      ____ H             ;start address of second program
TEMP:       DW       0000H              ;temporary storage for stack pointer

GO:         LXI      SP,STKB            ;startup routine — runs PRGMA first
            LXI      H,PRGMB            ;save fake return address as start
            PUSH     H                  ;of PRGMB
            PUSH     H                  ;save all registers in specific
            PUSH     D                  ;sequence
            PUSH     B                  ;
            PUSH     PSW                ;
            LXI      H, 0               ;save current SP in TEMP
            DAD      SP                 ;
            SHLD     TEMP               ;
            LXI      SP,STKA            ;set SP for PRGMA
            EI                          ;enable interrupts
            JMP      PRGMA              ;run PRGMA
;
; Upon receiving an interrupt, SWTCH changes program execution from
; current routine
;
SWTCH:      PUSH     H                  ;save all registers and return address
            PUSH     D                  ;on program's stack
            PUSH     B                  ;
            PUSH     PSW                ;
            LXI      H, 0               ;save current SP in TEMP after
            DAD      SP                 ;retrieving previous pointer
            XCHG                        ;
            LHLD     TEMP               ;
            XCHG                        ;
            SHLD     TEMP               ;
            XCHG                        ;
            SPHL                        ;new stack pointer
            POP      PSW                ;pop registers in opposite sequence
            POP      B                  ;
            POP      D                  ;
            POP      H                  ;
            EI                          ;enable interrupts after exiting
            RET                         ;SWTCH
;
;a JMP must be placed at the desired interrupt vector to transfer
;execution to SWTCH
;
            ORG      ____ H             ;desired interrupt vector (0038H if
INT:        JMP      SWTCH              ;vectored interrupt is not being used)
;
            END
```

*Program Listing.*

The following program was written for the 8080 series microcomputers, with or without vectored interrupt capability. In combination with a small amount of hardware, which is unnecessary if the system has a real-time clock, the program will allow simultaneous execution of two routines. It will also permit two users to share the same program, provided that it is reentrant coded.

The time-sharing routine, LINK, functions on an interrupt-timing basis. Each time an interrupt is received, LINK switches execution from Program A (PRGMA) to Program B (PRGMB) or vice versa. During the switch, all registers are saved, including the stack pointer, thereby isolating the two programs. However, care must be taken to insure that no variables are shared by the two programs as this will cause unpredictable errors.

For LINK to function properly, a pulse generator must be provided to generate periodic interrupts. As mentioned earlier, a real-time clock is suitable, provided it has an output in kHz. (A low-frequency oscillator output may be acceptable if there are no high-speed peripherals involved. It is also important to note that there is an upper limit to the switching frequency. Since

LINK must be executed for every switch, an equal number of instructions in the programs being run should be executed in order to maintain efficiency at about 25 percent per program.)

If there is no output of this frequency, or a real-time clock is not available, the circuit in Fig. 1 may be used. It consists of an NE555 timer operating in the astable mode to produce an output in the mid-audio range. This circuit may be connected directly to the +8 V line, provided it is adequately filtered. Otherwise, any voltage from +5 to +15 volts may be used.

The output from the timer or real-time clock should be connected to either PINT or one of the VI lines, depending on whether vectored interrupt is being used. If a coupling capacitor must be used, it should have a large capacitance, as a small value will degrade the quality of the generator's square wave.

As the program listing indicates, there are several operands left blank. These include the starting address of LINK, which can be anywhere within the available memory (LINK may be placed in ROM, provided the temporary variable, TEMP, resides in read-write memory), starting addresses for Programs A and B and the initial values of the stack pointer for the two programs. On page two, the address of the desired interrupt vector must be filled in. Afterwards, LINK may be assembled and loaded, along with the two programs to be run. Start-up is accomplished by applying power to the pulse generator, with interrupts disabled, and jumping to GO.

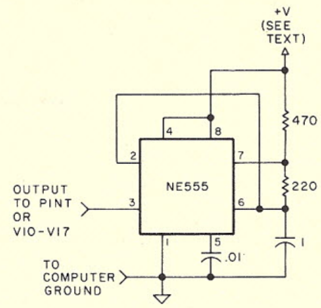


*Fig. 1.*

One final note: To facilitate testing the software, a debounced push-button switch to ground may be connected in place of the pulse generator (timer or clock). This will allow manual switching between the two programs.

It is my hope that extending the capabilities of the 8080 into the realms of time-sharing will defray the overall cost of microcomputer systems. ■

*Dave Maciorowski*
*173 Oakland Road*
*E. Pepperell MA 01437*

# Displaying Hexadecimal

## *and other related ideas*

Recently, while I was working on yet another peripheral for my microprocessor system, an interesting question came up: "What is the cheapest way to display hexadecimal digits on a panel display?" Hmmm. That should be pretty easy. Let's stroll over to the bookshelf and see what we can find.

The display type should probably be a seven-segment LED. They are plentiful and cheap, with many available for a dollar or so per digit. My favorite is the Hewlett-Packard 5082-7730/31. It is available in the MAN-1 style package and is bright and very readable at ten feet! It is a common-anode type, and a typical configuration for it is shown in Fig. 1(a). A diagram for common-cathode LEDs is shown in Fig. 1(b).

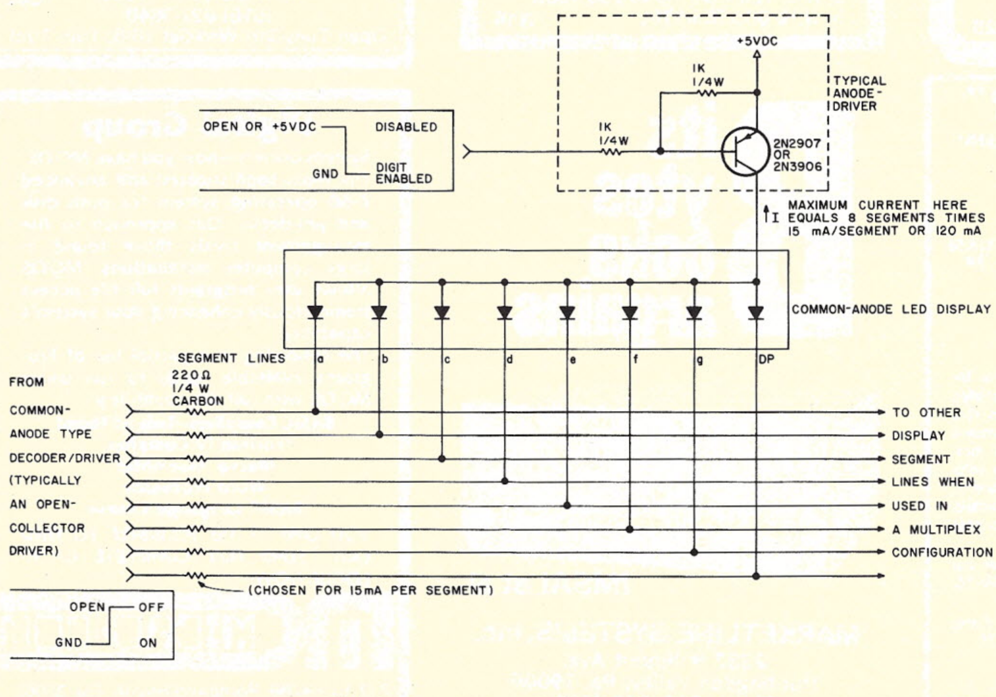



*Fig. 1(a). Common-anode display configuation.*

### What About Decoders?

The first logical place to look for decoders is in the 7400 logic series. These are plentiful and cheap (there are those key words again). Listed under "BCD-to-Seven-Segment Decoder/Drivers" we find the 7446, 7447, 7448 and 7449. These devices will decode a BCD input to the outputs necessary to activate a seven-segment display (see Fig. 2(a)). But look. Numerals 0 through 9 are OK, but A through F are nowhere to be found. (Instead we find gibberish.) A lot of good those are in displaying hexa-

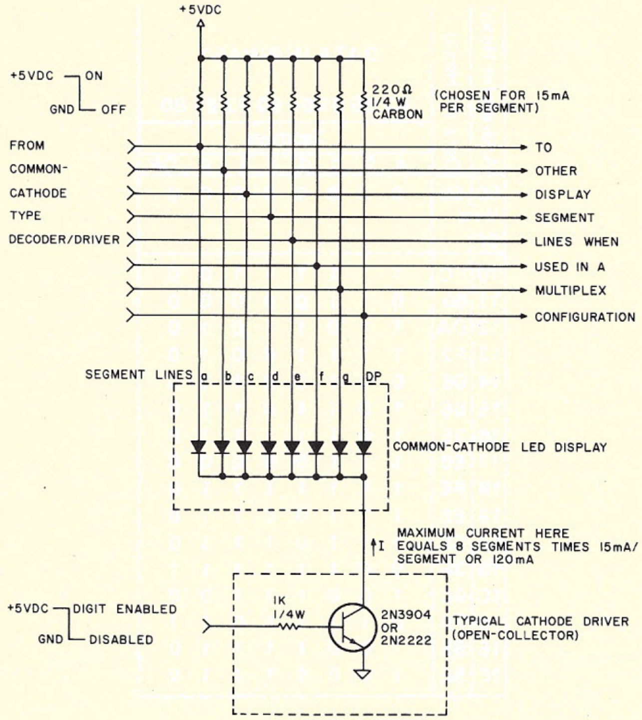



*Fig. 1(b). Common-cathode display configuation.*

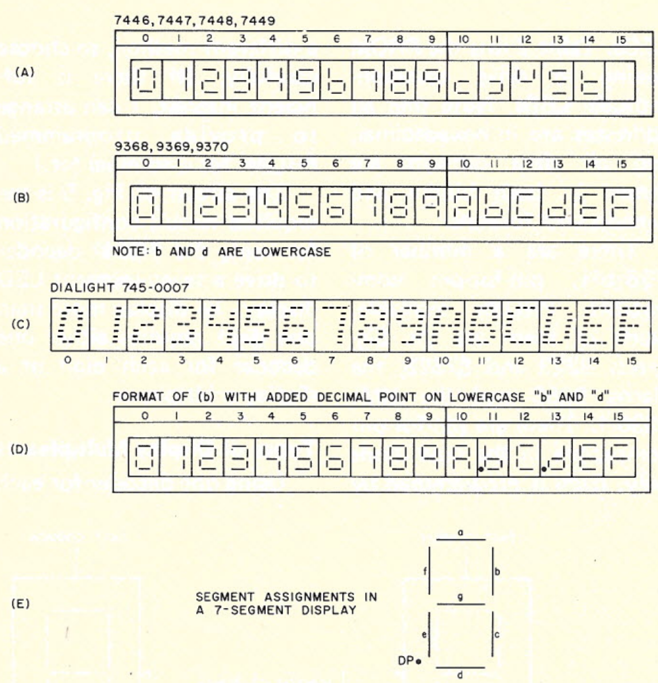



*Fig. 2. Display formats.*

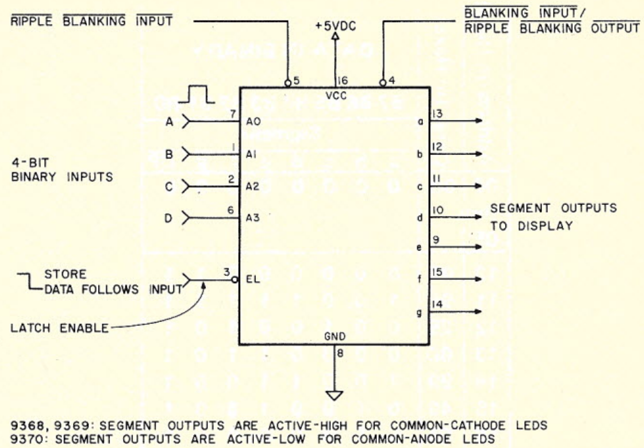



*Fig. 3.*

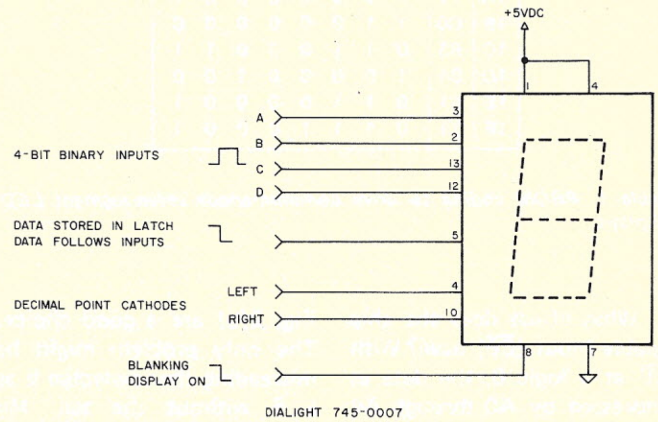



*Fig. 4.*

decimal digits. Well, what else can we find?

The Fairchild data book shows some interesting decoders. In addition to the ones above, there is a hexadecimal decoder/driver. Designated the 9368, 9369, 9370, these latch/decoder/drivers will display the digits of Fig. 2(b). The 9368 and 9369 drive common-cathode LEDs, while the 9370 drives common-anode LEDs. Typical circuit connections are shown in Fig. 3.

The Dialight 745-0007 is the next display we find. Its display format is shown in Fig. 2(c). It includes a latch, decoder, driver *and* hexadecimal LED display in one 14-pin dip package. Fig. 4 shows the pin connections. As my imagination conjures up a panel full of these, I come to a screeching halt at the price — $19 in single quantities. It's nice to dream, but that's a little out of my price range.

Gee, there doesn't seem to be much else available to do what we want. But all is not lost.

**Home Brewers, Take Note**

Why not *make* a decoder/driver? Sure. Use a PROM to convert a four-bit/binary input to the seven-segment outputs needed to drive a seven-segment LED display.

Let's take, for instance, the 82S23. It is a 256-bit Programmable Read Only Memory (PROM) arranged as 32 words of eight bits, and it has open-collector outputs, a good choice for driving LED displays (see Fig. 5). A 32-by-8 PROM has five address line inputs and eight data line outputs. Four of the five address lines (A0, A1, A2, A3) would be the four-bit binary inputs (A,B,C,D) needed. Seven of the eight data outputs would be the seven-segment driver outputs.

Can we utilize the other lines? Let's see. Address line A4 selects either the first or last 16 words of eight bits in the PROM. If we programmed the hexadecimal decoder in the second half of the PROM (A4 at a logic 1) and all zeros in the first half, then pulling A4 from a logic 1 to a logic 0 would cause all the segments to light, no matter what the code on A0 through A3 when used as a common-anode driver. Line A4 then becomes our lamp test input. When used as a common-cathode driver, this would cause all segments to shut off. Line A4 then becomes our blanking input.

| Address (in Hexa) | Data (in Hexa) | DATA IN BINARY | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| | | Segment | | | | | | | |
| | | a | b | c | d | e | f | g | DP |
| 00 thru 0F | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 03 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 11 | 9F | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 12 | 25 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 13 | 0D | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 14 | 99 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 15 | 49 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 16 | C1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 17 | 1F | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 18 | 01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 19 | 19 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1A | 11 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1B | C0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1C | 63 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1D | 84 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1E | 61 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1F | 71 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

*Table 1. PROM coding to drive common-anode seven-segment LED displays.*

| Address (in Hexa) | Data (in Hexa) | DATA IN BINARY | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| | | Segment | | | | | | | |
| | | a | b | c | d | e | f | g | DP |
| 00 thru 0F | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | FC | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 11 | 60 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 12 | DA | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 13 | F2 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 14 | 66 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 15 | B6 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 16 | 3E | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 17 | E0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 18 | FE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 19 | E6 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1A | EE | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1B | 3F | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1C | 9C | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1D | 7B | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1E | 9E | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1F | 8E | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

*Table 2. PROM coding to drive common-cathode seven-segment LED displays.*

What effect does the chip enable input ($\overline{CE}$) have? With $\overline{CE}$ at a logic 0, the data as addressed by A0 through A4 will be seen at the data outputs B0 through B7. When CE is raised to a logic 1, the outputs are disabled (i.e., all collectors are open). The chip enable input then becomes our blanking input when the PROM is used as a common-anode driver. When used as a common-cathode driver, this is the lamp test input.

Now, what do we program into the PROM? The digits of Fig. 2(b) are a good choice. The only problem might be misreading the lowercase b as a 6 without the tail. My choice is the digit format of Fig. 2(d). This uses the B0 data output line of the PROM to drive the decimal point in the LED display and turn it on for the lowercase b and d to clear up the ambiguity.

All PROM coding listed in the tables is for the display format of Fig. 2(d). Segments are assigned as in Fig. 2(e). Table 1 is the PROM coding to drive common-anode LEDs. Table 2 lists the PROM coding to drive common-cathode LEDs. Note that all addresses are in hexadecimal, and all data outputs are shown in both binary and hexadecimal formats.

There are a number of 256-bit, pin-for-pin compatible PROMs available. Some of them are the Signetics 8223 and 82S23, the Harris 8256 and the MMI 6330-1. These are pin-for-pin compatible in the read mode only. Each is programmed by a different method, so choose carefully. (If there is sufficient interest, I can arrange to provide programmed PROMs for a nominal fee.)

The circuit of Fig. 5 is the required circuit configuration for using the PROM decoder to drive a seven-segment LED display. To display more than one digit requires either one decoder for each digit or a display multiplexer.

**Types of Display Multiplexers**

Using one decoder for each

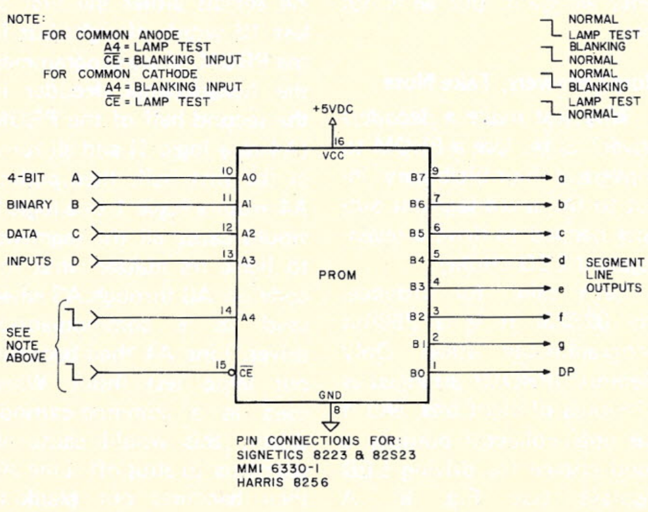



*Fig. 5.*

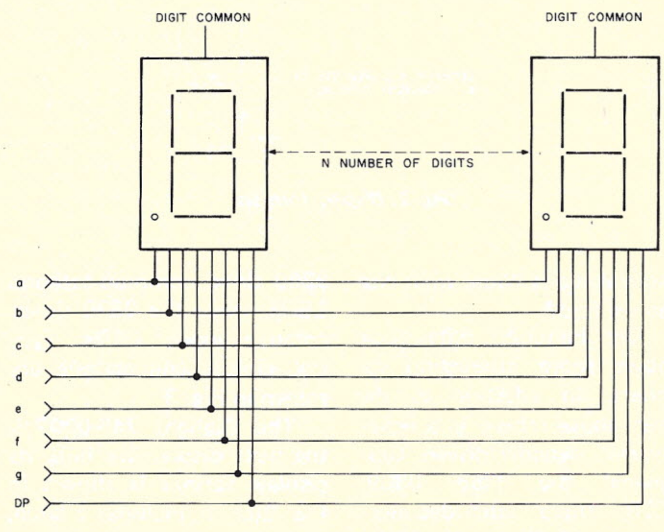



*Fig. 6. LED interconnection in a multiplexed display. Note: be sure to include current-limit resistors where required (see Fig. 1).*

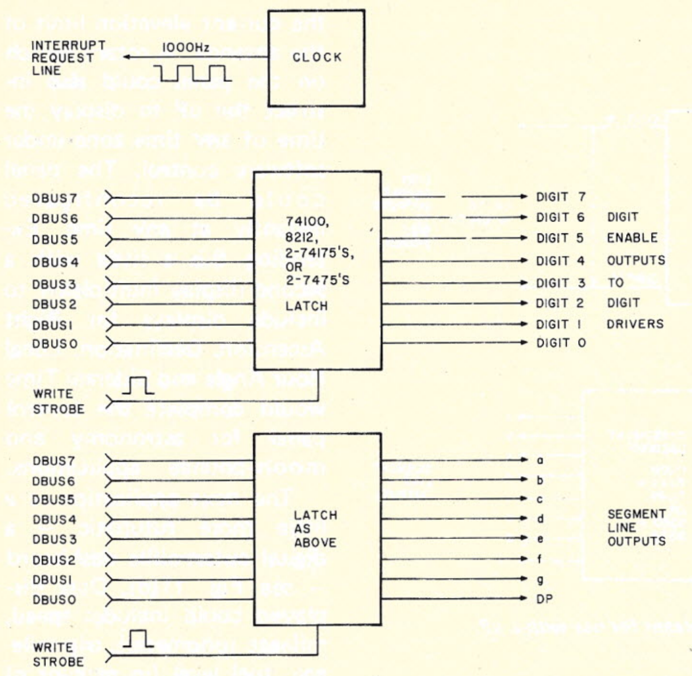



*Fig. 7. Minimal display multiplexer letting the uP do the multiplexing and binary-to-seven- segment conversion.*

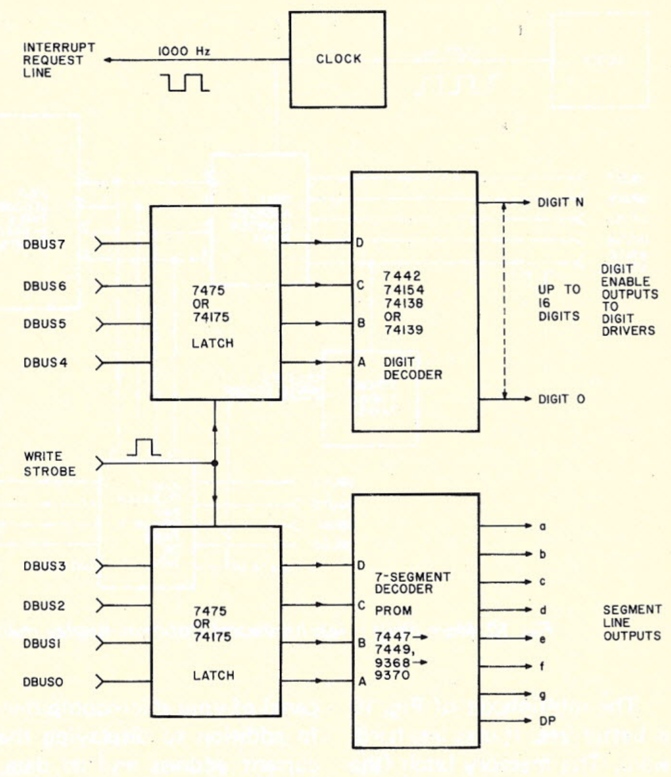



*Fig. 8. Adding the digit decoder and segment decoder in hardware cuts down software. uP still provides multiplexing function.*

digit is economical only up to two or three digits. For more digits than that, it is less expensive to use a display multiplexer. The multiplexer saves not only components, but, since only one digit is on at a time, the maximum power consumption is that of one digit and the supporting circuitry.

In the multiplexer configuration, the segment lines of all LEDs are connected in parallel (see Fig. 6). The resultant segment bus is then driven by the appropriate decoder/driver. Each digit is enabled at the same time that that digit's seven-segment information is placed on the segment bus. By sequentially enabling each digit with its associated data on a rotating basis, the display will appear to be on continuously. With this arrangement, each digit must be refreshed at least a thousand times a second, i.e., at a 1000 Hz rate, to prevent a flickering display.

The simplest type of display multiplexer for use with a microprocessor (uP) would let the uP do the multiplexing *and* the binary-to-seven-segment conversion in software (see Fig. 7). The software conversion would reduce the hardware requrements but would increase the size of the software handler and the time needed to execute it. Note the clock connected to the interrupt line. The interrupt would cause the uP to jump to a subroutine that would service the display. It would change the digit address and its associated data at the 1000 Hz rate.

The next step in hardware complexity moves the binary-to-seven-segment conversion from software to hardware (see Fig. 8). This is where our PROM comes in. The same interrupt sequence would be used here, except that the actual four-bit binary data to be displayed would be loaded into the data latch instead of the converted seven-segment information.

The next type of multiplexer is the simplest from a software point of view (see Fig. 9). There is no refreshing necessary by the uP. All the uP need do is write the four-bit binary data into the appropriate data latch. The hardware then performs the multiplexing function.

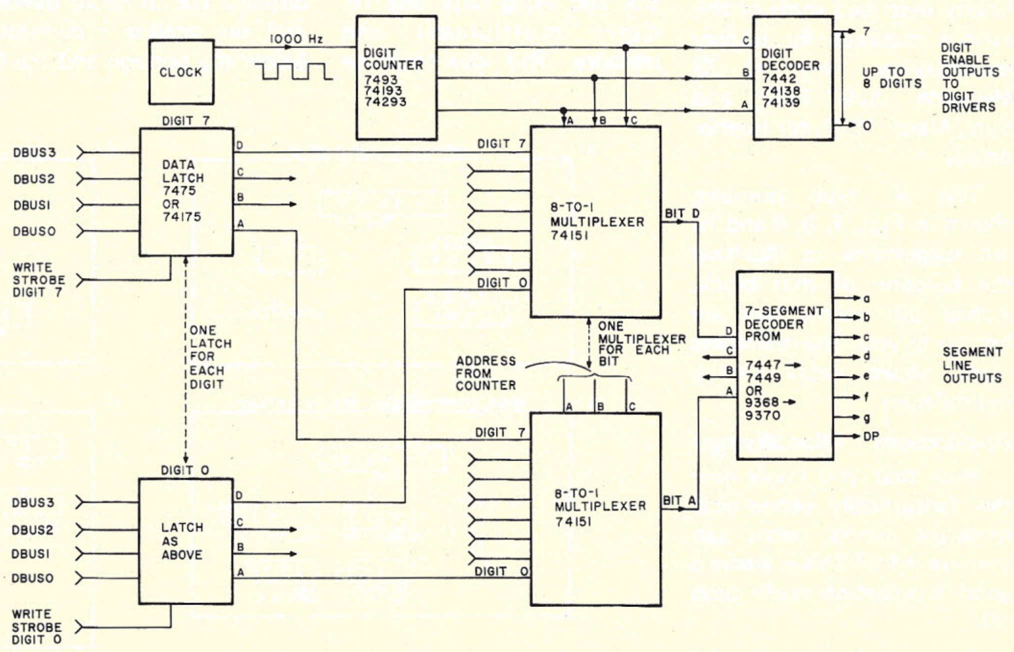



*Fig. 9. Hardware multiplexer requires no refresh from uP.*

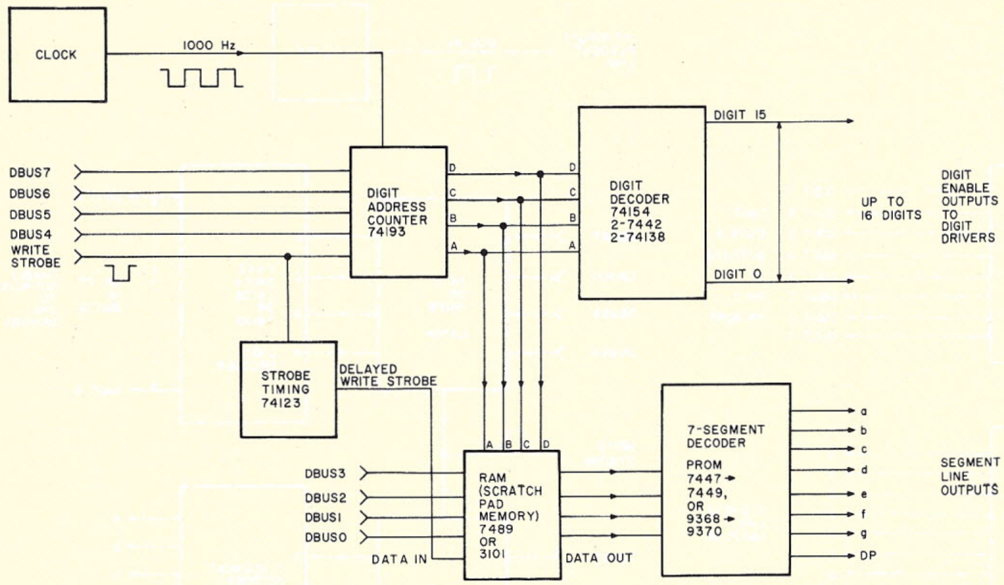



*Fig. 10. More digits — less hardware! Optimum display multiplexer for use with a uP.*

The multiplexer of Fig. 10 is better yet. It uses less hardware. The memory latch (the 7489 or 3101 scratchpad memory), arranged in 16 words of four bits, will *store and multiplex* 16 digits of information all in one 16-pin package. This circuit is especially nice for driving from a uP's eight-bit parallel port and can be used for any number of digits up to 16.

Actual construction details for display multiplexers have been adequately covered in recent literature and will not be repeated here. So, I only briefly described some of the various methods for display multiplexing. Refer to *73 Magazine*, June 1977, and *Byte*, March 1977, for further details.

The IC type numbers shown in Figs. 7, 8, 9 and 10 are suggestions to illustrate the function of that block. Actual pin connections are left up to you. The references made above detail several multiplexers.

## Applications (Blue-Skying!)

Now that you know how this fantastically economical hardware works, what can you use it for? This is where a good imagination really pays off.

Look at Fig. 11(a). This could be the intelligent front panel of your microcomputer. In addition to displaying the current address and its data, you get an auxiliary display of 16 bits (four digits) and a six-digit real-time clock display using only one 16-digit display multiplexer.

How about the panel shown in Fig. 11(b)? Here, all in one place in an easy-to-read format, is temperature (from any number of sensors) in either Celsius or Fahrenheit (or Kelvin?), percent of relative humidity, wind speed and direction, and a real-time clock all under processor control and using only one 16-digit multiplexer and memory. This idea could be expanded to include the monitoring of the efficiency of your solar heating system!

Fig. 11(c) could be the status panel for an amateur radio operator who works through the Oscar satellite or receives weather data from the NOAA weather satellites. Here the displays show Universal Coordinated Time (UTC or GMT), the amount of time since the satellite crossed the equator (in minutes and seconds) and the azimuth and elevation of the antenna to work through the satellite. Note the .b0 on the elevation display. The .b could denote that the satellite is currently below the horizon and the 0, the current elevation limit of the antenna. A rotary switch on the panel could also instruct the uP to display the time of any time zone under software control. The panel could be reconfigured instantly at any time. Extending this a little with a second display multiplexer to include displays for Right Ascension, Declination, Local Hour Angle and Sidereal Time would complete the control panel for astronomy and moon-bounce applications.

The next application is a little more futuristic — a digital automobile dashboard — see Fig. 11(d). Data displayed could include: speed, mileage (odometer), trip mileage, fuel level (in percent of capacity), temperature (of engine, transmission, rear axle), engine oil pressure, alternator current and a real-time display of miles per gallon. Of course, time would be included (with the switch to choose the time zone) and another switch could select readings to be in either the English or metric numbering system.

I hope I've jogged your imagination into high gear — there are more real-time applications for this small amount of hardware (for $25 or less) than any one of us could possibly conjure up. Send them in so we can all benefit. Happy computing! ■

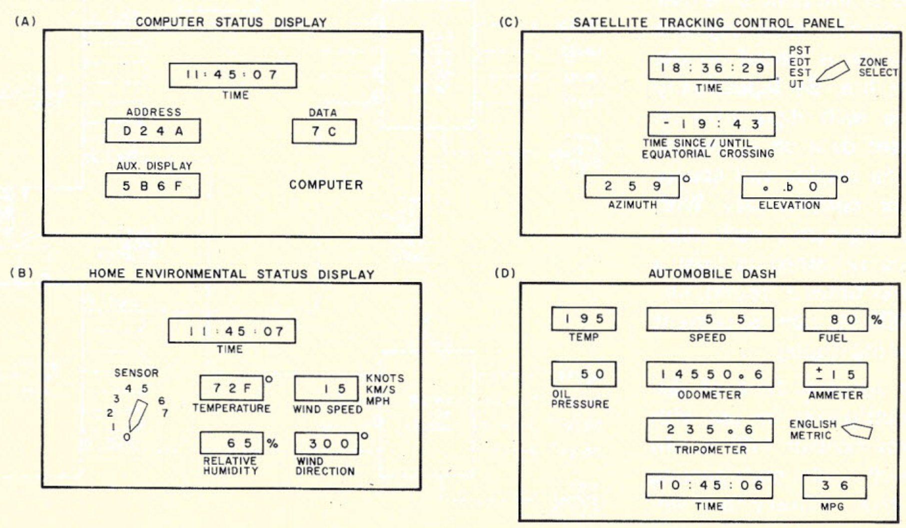
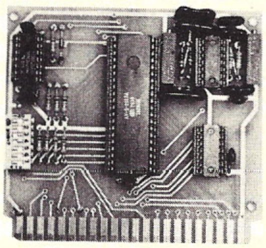



*Fig. 11. Application ideas.*

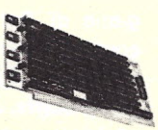
# ELECTRONIC SYSTEMS
## P.O. Box 9641 San Jose CA 95157
## (408) 374-5984



## UART & BAUD RATE GENERATOR

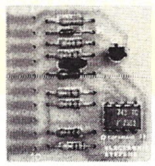
Part no. 101
- Converts serial to parallel and parallel to serial
- Low cost on board baud rate generator
- Baud rates: 110, 150, 300, 600, 1200, and 2400
- Low power drain +5 volts and -12 volts required
- TTL compatible
- All characters contain a start bit, 5 to 8 data bits, 1 or 2 stop bits, and either odd or even parity.
- All connections go to a 44 pin gold plated edge connector
- Board only $12.00; with parts $35.00

## 8K STATIC RAM



Part no. 300
- 8K Altair bus memory
- Uses 2102 Static memory chips
- Memory protect
- Gold contacts
- Wait states
- On board regulator
- S-100 bus compatible
- Vector input option
- TRI state buffered
- Board only $22.50; with parts $160.00

## RS-232/TTL INTERFACE



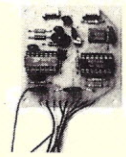
Part no. 232
- Converts TTL to RS-232, and converts RS-232 to TTL
- Two separate circuits
- Requires -12 and +12 volts
- All connections go to a 10 pin gold plated edge connector
- Board only $4.50; with parts $7.00

## DC POWER SUPPLY

Part no. 6085
- Board supplies a regulated +5 volts at 3 amps., +12,-12, and -5 volts at 1 amp.
- Circuit has filters, rectifiers, and regulators.
- Power required is 8 volts AC at 3 amps., and 24 volts AC C.T. at 1.5 amps.
- Board only $12.50

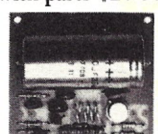
## TIDMA

Part no. 112
- Tape Interface Direct Memory Access
- Record and play programs without bootstrap loader (no prom) has FSK encoder/decoder for direct connections to low cost recorder at 625 baud rate, and direct connections for inputs and outputs to a digital recorder at any baud rate.
- S-100 bus compatible
- Board only $35.00; with parts $110.00



Part no. 111
## TAPE INTERFACE

- Play and record Kansas City Standard tapes
- Converts a low cost tape recorder to a digital recorder
- Works up to 1200 baud
- Digital in and out are TTL-serial
- Output of board connects to mic. in of recorder
- Earphone of recorder connects to input on board
- Requires +5 volts, low power drain
- Board $7.60; with parts $27.50
- No coils



Part no. 107
## RF MODULATOR

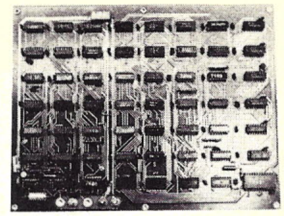
- Converts video to AM modulated RF, Channels 2 or 3
- Power required is 12 volts AC C.T., or +5 volts DC
- Board $7.60; with parts $13.50

## Apple II Serial I-O Interface

Part No. 2
- Baud rates up to 30,000
- Plugs into Apple Peripheral connector
- Low-current drain
- RS-232 Input and Output

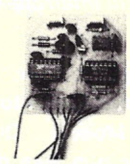
SOFTWARE
- Input and Output routine from monitor or BASIC to teletype or other serial printer.
- Program for using an Apple II for a video or an intelligent terminal. Board only – $15.00; with parts – $42.00; assembled and tested – $62.00.

## TELEVISION TYPEWRITER



Part no. 106
- Stand alone TVT
- 32 char/line, 16 lines, modifications for 64 char/line included
- Parallel ASCII (TTL) input
- Video output
- 1K on board memory
- Output for computer controlled curser
- Auto scroll
- Non-distructive curser
- Curser inputs: up, down, left, right, home, EOL, EOS
- Scroll up, down
- Requires +5 volts at 1.5 amps, and -12 volts at 30 mA
- Board only $39.00; with parts $145.00

## MODEM



Part no. 109
- Type 103
- Full or half duplex
- Works up to 300 baud
- Originate or Answer
- No coils, only low cost components
- TTL input and output-serial
- Connect 8 ohm speaker and crystal mic. directly to board
- Uses XR FSK demodulator
- Requires +5 volts
- Board $7.60; with parts $27.50

## To Order:

Mention part number and description. For parts kits add "A" to part number. Shipping paid for orders accompanied by check, money order, or Master Charge, BankAmericard, or VISA number, expiration date and signature. Shipping charges added to C.O.D. orders. California residents add 6.5% for tax. Parts kits include sockets for all ICs, components, and circuit board. Documentation is included with all products. Dealer inquiries invited. 24 Hour Order Line: (408) 374-5984.

VISA master charge

E21

*Ken Barbier*
*Borrego Engineering*
*PO Box 1253*
*Borrego Springs CA 92004*

# Build a Touch-Response Display

## *an advance in human engineering*

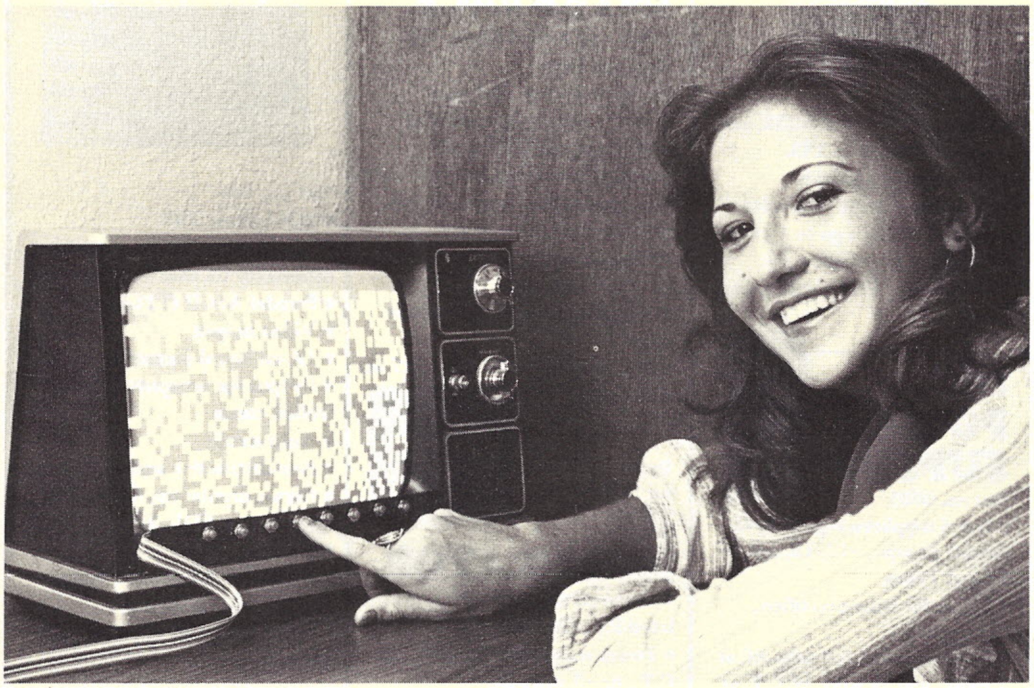


*Photo 1. The computer idles, producing a random pattern of black-and-white spaces. Pressing any key will interrupt this background display and call up a game display.*

Captain Kirk was alone, strapped into the Exploration Module. On a screen before him was displayed the star field through which he was traveling.

Suddenly, in the upper right corner of the display, a bright circle appeared around one of the dots of light. The on-board computer had detected an image that didn't fit the characteristics of any known celestial object. Kirk placed his fingertip lightly within the circle and moved it to the center of the screen. The image followed his finger. When the circle was centered in the screen, he drew a small box around it with his finger. Instantly that area was expanded to fill the entire screen. Sure enough, the dot could now be identified as a Klingon war cruiser.

At the lower edge of the screen, several computer command legends were displayed. Kirk touched three of them in turn: RED ALERT, HOSTILE VESSEL and EMERGENCY RETURN. Then he sat back and relaxed as the on-board brain relayed his red-alert command and the identification and coordinates of the intruder back to the Enterprise. The computer then automatically reversed the module's course at maximum speed to return to the safety of the starship.

*Photos by Dave Rosenbush.*

*Computer operator: Louise Kuelimer.*

### Back to the Real World

This scene would never appear on the TV show. Not enough shouting and arguing and human activity. Too many actions can be accomplished in too short a time for the viewers to follow. It's hardly TV material.

Back in the real world, touch-response displays have been developed and may shortly appear on the market. But not at a hobbyist budget level. A transparent plate in front of a CRT containing a grid of fine wires (similar to the X and Y address lines of a core memory plane) can detect the electrical noise produced by a fingertip touching the intersection of two wires. The computer, knowing what it has displayed at that location, can then determine that the human operator has made that particular selection. One such selection can then call up another display, with a new set of options for the operator. This touch-response display technique could find unlimited application. We will mention a few later.

Light pens are available, of course, to enable an operator to designate a particular spot on a CRT display. But light pens are not inexpensive and require extensive support hardware to translate the time of occurrence of a flash of light (as the raster-scanned bright dot passes the pen) into meaningful screen coordinates. Knowing that neither technique would be within my hobby budget, but wanting some form of man-machine communication that a child or non-typist could use, I developed the (almost) touch-response TV display described below.

### Using the Touch-response Keys

Louise is ready to play a game of Battleship, with my computer as an opponent. As shown in Photo 1, before the game starts, a random pattern of black-and-white spaces is displayed on the TV screen. This pattern is generated by the game program random number generator and is updated several times a second to produce a constantly changing display. This keeps the random number generator running so that it doesn't start at the same place every game, and also prevents any static display image that could burn itself into the phosphors on the face of the TV tube. (Software random number generators are not truly random, and will produce the same sequence of numbers over and over.) To begin play, Louise will press any one of the eight push-button switches attached to the bottom of the TV screen.

The eight switches are connected to the computer through a single 8-bit input port, as shown in Fig. 1. The Intel 8080 program listing (Program A) is a subroutine that will return an 8-bit image of the eight switches to the calling program whenever a switch is pressed and released. Using software to provide switch debounce simplifies the hardware.

Having interrupted the background display, and therefore informed the computer that Louise is ready to try to search out a hidden battleship, she is presented with the display shown in Photo 2. An eight-by-eight cell grid is formed by vertical columns of Is and horizontal rows of underlines. Each cell is identified by a numbered X coordinate and a lettered Y coordinate. Somewhere in this grid of 64 cells is hidden a battleship consisting of three cells in a vertical, horizontal or diagonal line. Louise decides to try firing a shot at the cell identified by an X coordinate of 5, Y coordinate of C.

At this point, it would certainly be nice to enable her to select this spot by touching it on the screen. At the other extreme, we could compel her to search out the characters 5 and C on a keyboard in the correct sequence. If she were not a typist this alternative would not make any points with her; she might soon tire of this and any other games we had in mind. That wouldn't do!

In Photo 2, the computer is asking her to select an X coordinate from among the choices displayed above the row of switches. No chance for confusion here, as the switch legends displayed correspond exactly with the identifications displayed above each column of the grid. So Louise presses the 5 switch.

Instantly the display changes to that shown in

```
PORT    EQU     ----      ; SET TO PORT ADDRESS
        ORG     ----      ; SET TO START LOCATION
PBSW    IN      PORT      ; ANY SWITCH PUSHED?
        CMA               ; INVERT BITS
        ANI     OFFH      ; TEST FOR ALL ZEROS
        JZ      PBSW      ; NO ONES, WAIT
        MVI     A,OFFH    ; GOT ONE, SET UP
PBSW1   DCR     A         ;      A DELAY LOOP
        JNZ     PBSW1
        IN      PORT      ; READ SWITCHES AGAIN
        CMA
        ANI     OFFH      ; STILL PUSHED?
        JZ      PBSW      ; NO, START ALL OVER
        PUSH    PSW       ; YES, SAVE DATA
PBSW2   IN      PORT      ; WAIT FOR END OF
        CMA               ;      SWITCH CLOSURE
        ANI     OFFH
        JNZ     PBSW2
        MVI     A,OFFH    ; AND DELAY AGAIN
PBSW3   DCR     A
        JNZ     PBSW3
        POP     PSW       ; RESTORE SWITCH IMAGE
        RET               ; AND RETURN
        END
```

*Program A. Switch Read Subroutine. Written in Intel 8080 assembly language, this subroutine will read the switches, provide switch debouncing with a delay loop and return to the calling program with an image of the switches in the A register.*

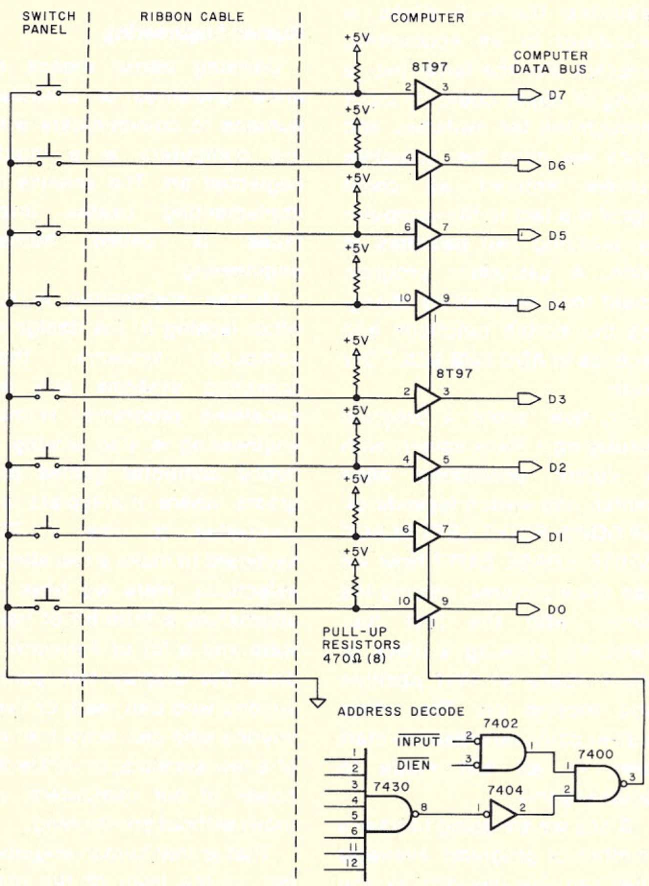



*Fig. 1. Schematic diagram. The normally open push-button switches are input to the data bus "low true," so the A register will have to be complemented. The inputs to the 7430 will be connected to the address bus directly or through inverters to set up a particular port address.*

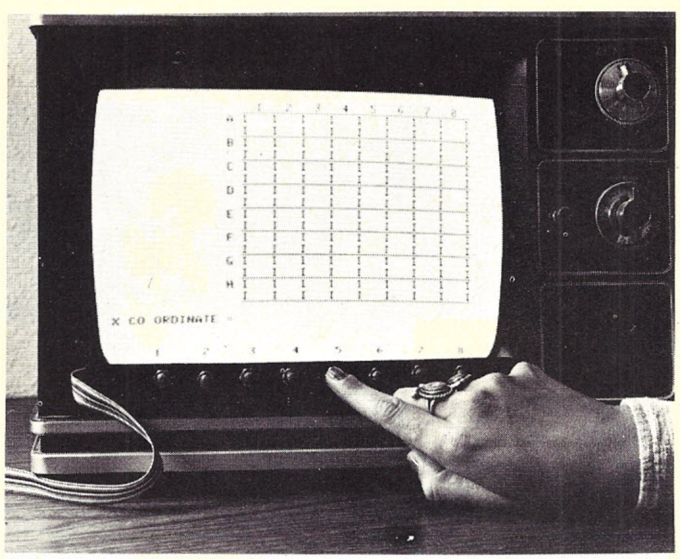


*Photo 2. The operator is asked to enter the horizontal (X) coordinate of the cell he or she wishes to shoot at. The switch legends correspond to the positions along the X axis.*

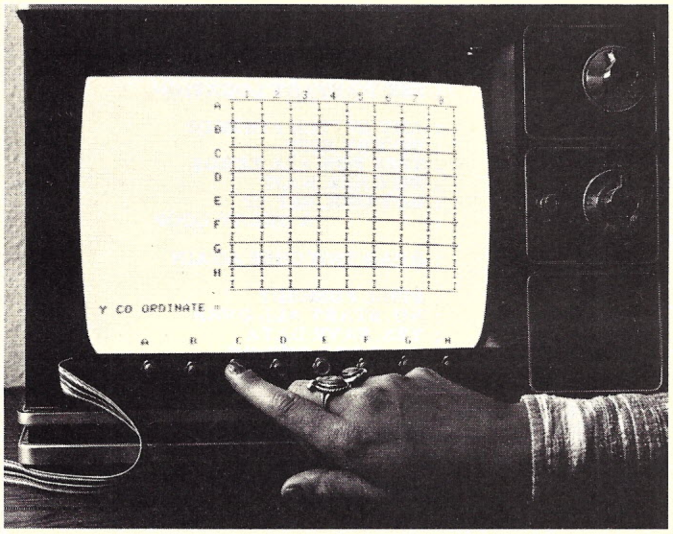



*Photo 3. The Y coordinate is requested, and the switch legends are changed accordingly.*

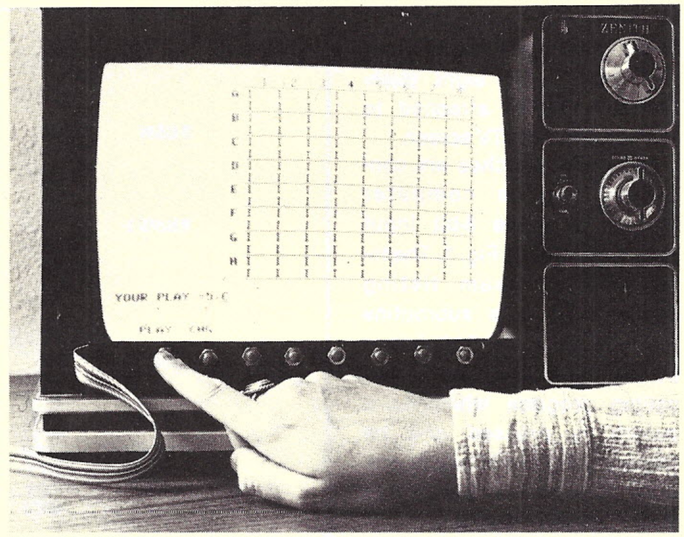



*Photo 4. The X and Y selections are displayed, and the player has the option of entering the shot or requesting the opportunity to change the selection. Pressing CHG (for change) would cause a repeat of the display shown in Photo 2.*

Photo 3. She is now asked for the Y coordinate, and the switch legends have been changed to the letters (A to H) corresponding to the identifications on each row of the grid above. So she presses the C switch.

Just in case she might have changed her mind, or pressed the wrong switch, the computer will give Louise a chance to take back her move. Her choices are shown in the next display (Photo 4), and she has the option of entering the play or changing it. If she presses the CHANGE switch, the program reverts back to the display of Photo 2, and she can start this move over again. If she is happy with her selection, she presses PLAY, and the computer records her move as either a hit or a miss.

The display shown in Photo 5 is a later stage in another battleship game, with a hit shown in cell 6/F, and a miss (the light shading) shown in cell 2/C. The game will continue until three hits in a row are recorded. At this point, the switch options NEXT PLAYER, or NEW GAME?, or DONE?, or whatever your program would require as the next operator input, could be programmed into the game.

Battleship on an 8-by-8 grid is a rather trivial game and is used here only to demonstrate the technique of the touch-response display. A million ideas are probably already springing to life in our brains—if time would only permit their development! Checkers and chess obviously fit the 8-by-8 grid. Expanding the number of switches to ten would permit the input of decimal digits to any program requiring them—a game, a calculator or an accounting program. For the latter two, a string of digits could be input through the ten switches, and when we have the complete number entered we could signal the fact to the computer by pressing *two* switches at once. A calculator program could then respond by changing the switch functions and legends to ADD SUB MULT DIV = etc.

Or, how about a program displaying a blank screen, with a cursor positioned dead center, and switch legends of: UP DOWN RIGHT LEFT BLACK WHITE ERASE EXIT? Now we can draw pictures, moving the cursor with the first four switches, drawing a black or white space at that position and moving on. The whole screen could be erased to start over, or an exit made to another game.

Since we are going to have a number of programs available that can use this display and switches, perhaps the first display following the random background should be a list of available programs, with each key assigned to a different program. Then we could call up Battleship, or Calculator, or Checkers, or Chess, or Ticktacktoe. The options are limited only by our imaginations.

**Human Engineering**

Devising better means to allow untrained or unskilled humans to communicate with our computers is a much-neglected art. The science of implementing usable interfaces is called human engineering.

Human engineering is too often lacking in the design of computer systems, their operating systems and applications programs. Human engineering is also lacking in hobby computer games programs where non-typists are compelled to use a TTY keyboard to make a few simple selections. Here we have an alternative: a little bit of hardware and a lot of software to drive the display will permit anyone who can read, or even anyone who can learn the use of a few symbols, to utilize the power of our computers, virtually without any training.

That is real human engineering, at the level of the man-machine interface. And who knows? The human engineering groundwork we lay today may someday enable Captain Kirk to escape the Klingons! ■

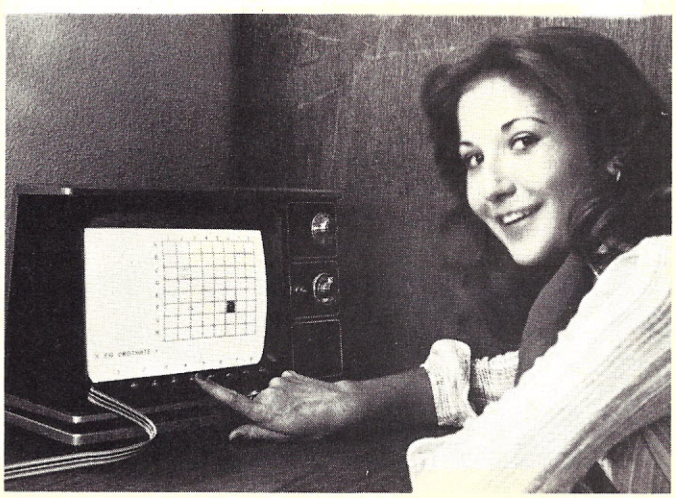


*Photo 5. Two shots have been entered. The one at cell 6/F is seen to be a hit. The shot at cell 2/C missed the target.*

*For KILOBAUD readers who have experienced difficulties with* **Associated Electronics** *or had problems contacting them, their address is:*

**ASSOCIATED ELECTRONICS**
**12444 Lambert Circle**
**Garden Grove, CA 92641**
**(714) 539-0735**

*Dave Lien*
*8662 Dent Dr.*
*San Diego CA 92119*

*Dave Waterman*
*834 Oak Lee Ln.*
*Alpine CA 92001*

# Turn It Off!

## *power-down mod for the TRS-80*

*This article details a simple, but useful, modification to the TRS-80. One of the authors, Dave Lien, will soon be contributing regularly to a column in* Kilobaud *about the TRS-80. —John.*

It is in the best tradition of experimenting that owners of electronics equipment (or most anything) are not content with something the way it left the factory. With the advent of high-priced commercial equipment, experimenters have become increasingly reluctant to make changes. Here, however, is a nice simple modification to the Radio Shack TRS-80 video monitor that's almost impossible to goof up. It's an ideal "first" foray into the computer hardware thicket. Besides being a useful and inexpensive modification, it helps relieve the fear of peeking into the box to see what you've bought.

### Why Mess with Success?

As furnished, the TRS-80 requires three 120 V outlets, one of them polarized. Once the external power supply is plugged in it stays on forever, using a small amount of electricity even when the computer is turned off. The computer's on/off switch does not turn off the power supply. People concerned with turning everything completely off when leaving the house, particularly considering the unhappy history of some "instant on" TV set fires, will want to have a handier way to turn off the entire computer system than having to push switches and unplug components. As luck would have it, the TRS-80 can be easily modified for single-switch on/off power control, the pushbutton switch on the monitor controlling the entire system (see Photo 1). It costs less than a dollar and takes 15 minutes to make operable.

### Scalpel . . .

Unplug everything. Remove the five screws from the back cover of the video monitor, and carefully pull off the cover. Using a nibbling tool, drill and pocket knife, blowtorch, jackhammer or whatever is handy, cut a neat hole in the cover to fit a single 120 V chassis mount socket. Position the socket as shown so a three-port cube tap will fit, with good clearance in all directions.

It's important to note that there are two common cube-tap configurations. Look carefully at the relation between the plug tangs and the socket slots on the cube tap shown in Photo 2. The other kind of tap has the plug and sockets offset 90 degrees from each other. Bet you never noticed. The local drug store, grocery or harware store will generally carry both kinds. Obviously, the other kind would not allow all three tap outlets to be accessible with the socket positioned as it is.

### Soldering Iron . . .

Mount the socket in place with two machine screws. Cut a 12 inch piece of ordinary lamp cord, and strip about 3/8 inch of

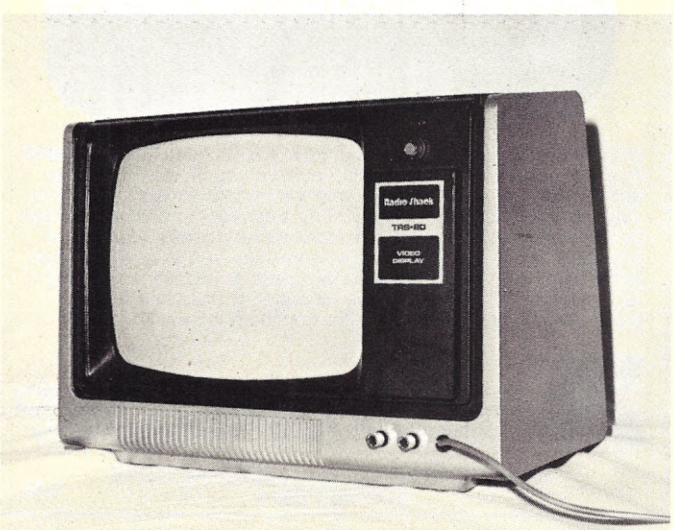



*Photo 1. Radio Shack TRS-80 video display monitor.*

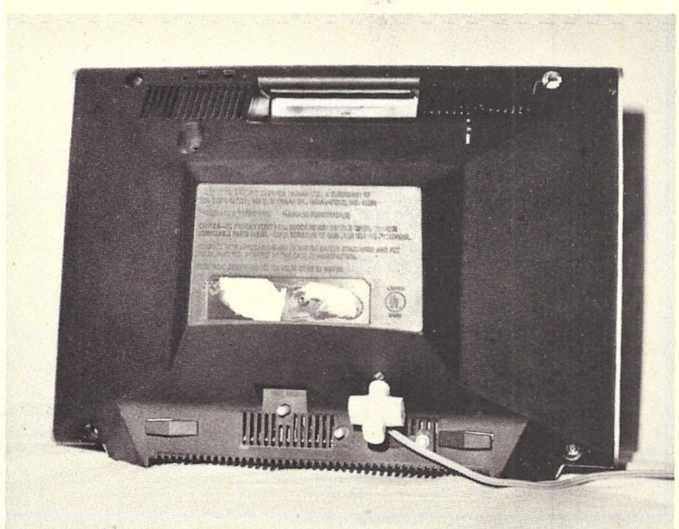


*Photo 2. Back panel of TRS-80 with cube tap installed.*

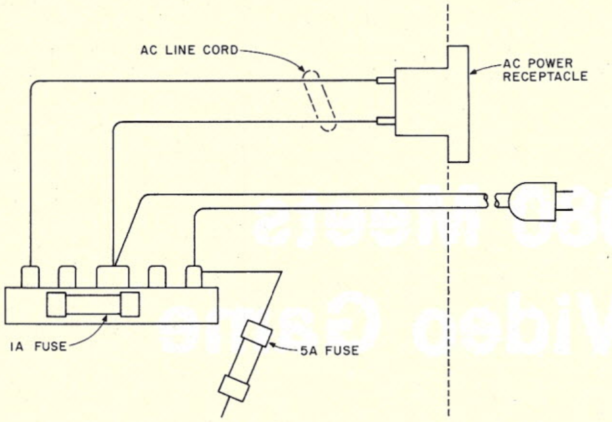



*Fig. 1. Schematic for power-switch modification.*

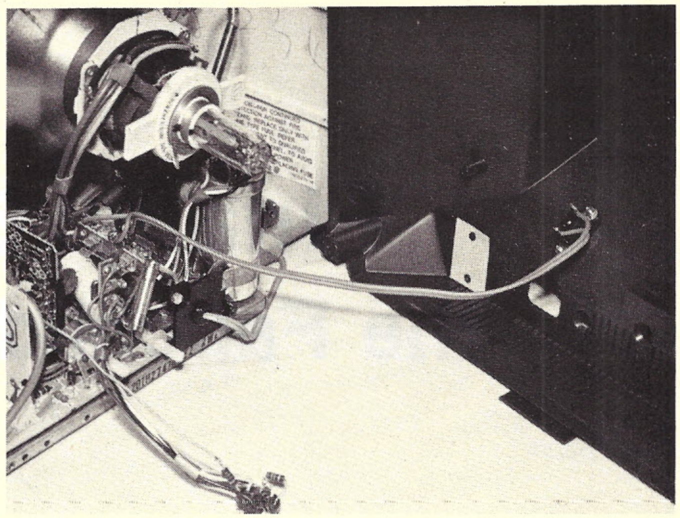


*Photo 3. The picture-tube socket is unplugged here to make the wiring easier to follow. It is not necessary to unplug it to make the modification.*

insulation off both wires on both ends. Tin all four ends, assuming you want to do this job right.

Solder the two wires on one end to the terminal lugs, and the other ends of the two wires to the receptacle as shown in Fig. 1 and the close-up photo in Photo 3. Before you put the back cover in place, insert the four mounting screws into the sockets at each corner of the lid. (See how easy this is.)

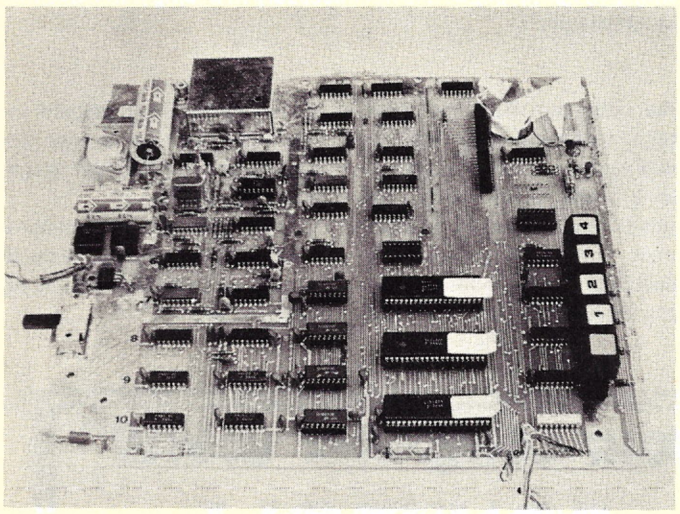
Tuck the extra length of lamp cord out of the way and fit the back cover into place. Tighten the four corner screws and replace the back chassis-mount screw.

**Putting it to Work**

Plug the cube tap into the new outlet in the back cover. Plug the TRS-80 power supply and the cassette recorder into the cube tap. Plug the monitor's polarized plug into the wall outlet. Push the computer's power switch to the ON position. It will be left on since now everything will be switched at the monitor. Turn the monitor on and watch for the red light on the computer to light, the monitor screen to light up and the recorder to run when you tell it to.

Once you use this simple one-switch hookup, you'll wonder why you didn't think of it sooner.■

# Finally: 8080 Meets the Fairchild Video Game



The Innards. *Just look at all the goodies. Forty TTL and CMOS ICs along with a bunch of other neat stuff including a DIP resistor package, some push-button switches, regulators, rf oscillator (in the shield), 3.579 MHz crystals, etc. Everything plugs into filters and connectors.*

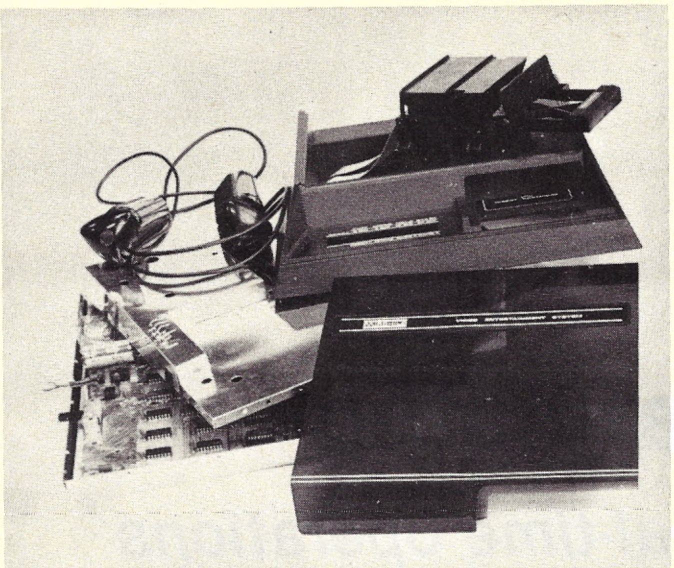
*Jim Huffman*
*Hufco*
*PO Box 357*
*Provo UT 84601*

I recently purchased one of Fairchild's F8 Microprocessor video games. I thought you'd be interested to know what we found here at HUFCO by dissecting my unit and through several conversations with various people at Fairchild. Amazingly, we found that it is possible to use the Fairchild F8 Microprocessor controlled video game as a color video display for some pretty impressive graphics. In fact, they are interfacing one to an 8080 bus right now and will release full conversion data in a month or so. The insides of this $150 video game are very impressive. There's an F8 Microprocessor chip and two PSUs (Program Storage Units) or factory mask programmed ROMs. They contain the operating system and two video games which are supplied (built into the basic unit). The operating system and the two games are distributed within the chips so that it would be impossible to replace one of the PSUs with another which contained new video games. There were a couple of sockets on our unit. These were used for testing and production checkout. Also, our particular unit (an early model) had a few strange interconnections on it. In several places the PC tracks had been cut and rewired and in one place the track had been cut and a jumper wire which bridged several other tracks was added. Additional ferrite filtering was included on the +5 and +12 lines. The filters had been added by breaking the existing PC tracks. A tip — be careful when looking around in there — there are CMOS circuits as well as 74LS and standard TTL circuits. Static damage could result to the CMOS circuits.

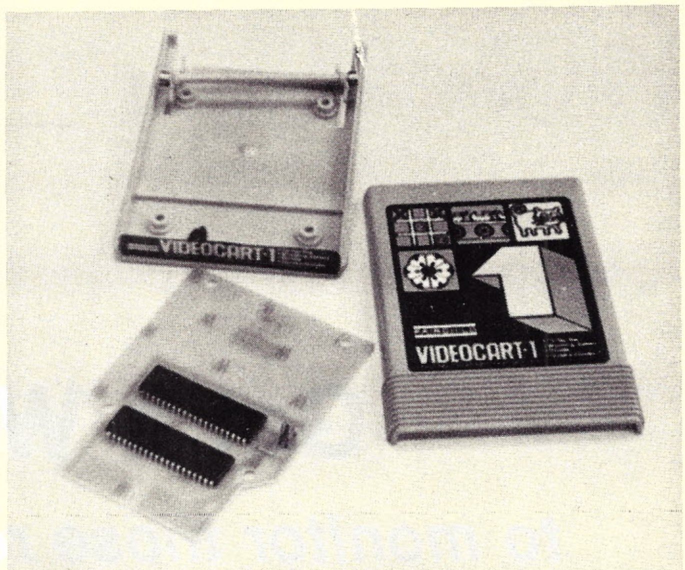
The dissected unit is shown in the photo. There were paper tags on the F8 Microprocessor and the two PSU chips so that we could keep track of which was which as we traced the interconnecting wiring. At first glance, we thought we had a three chip system, one with the static memory interface IC. We found the power pin interconnections were such that there is a single CPU chip and the two PSUs. Behind the program storage units are the 4K dynamic RAMs used for display refresh. They are Fairchild number 9023, a number that is not yet given in any Fairchild data books. Two memory chips on two planes are used to decode color. Only 6K of the remaining 8K are used to create the actual display; 2K are wasted.

The next point of interest is shown on top of the stack in the photo. This card-reader-like device is the cartridge interconnect that allows the cartridge held video games to be plugged into the main unit. There are enough pins available to the experimenter to allow interface of more PSU and static memory interface chips for adding external memory or for interfacing to a memory slot on a processor. A look at the outputs that are available through the cartridge interconnect will give you an idea of the great potential of this video game. There are eight data lines, five ROM control lines, an IRQ interrupt, two clock terminals ($\Phi$ and write) and finally, +5, +12, and ground.

Doing something other than playing video games with this video game should be a piece of cake. For a stand-alone system, a Fairbug/Fairchild operating system stored on a PSU is available for only $13.90 in single quantities from your nearest Fairchild distributor. Using the program listing as in Fairbug there would be an I/O assignment conflict, i.e., Fairbug uses I/O ports four and five, and so does the video entertainment system. You could add the static memory interface and some I/O level translators (such as



The pile. *Here are all the major subassemblies of the main unit. From its appealing smoked Plexiglas cover to the two 8-direction control handles, the unit has class. It looks just as good on the inside, too.*




A Videocart undressed. *Here is one of the famous videocarts with its two ROMs containing four games. It has a small spring door to keep the contact fingers away from human fingers.*

converting RS232 to TTL) and you would be able to have a stand-alone full F8 Microcomputer system.

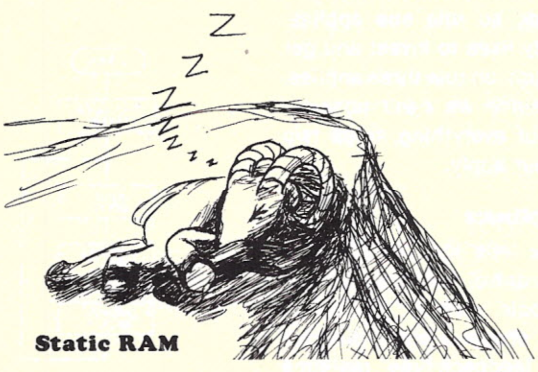
For a color video display circuit look to the Xetron division of Fairchild. They are the ones responsible for developing the software games. They also have in their possession complete information for interfacing the F8 video game to an 8080 microprocessor system. Obviously, enterprising 6800 owners could also use it.

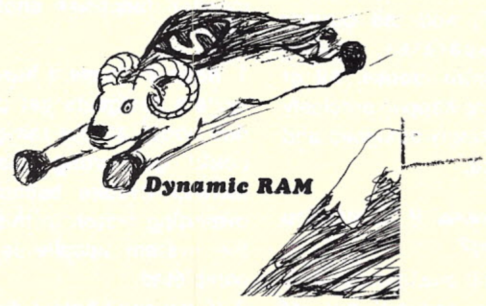
The graphics display is color with an approximate 96 x 64 display matrix. If you've ever seen one of these babies in action, you know about its capabilities. It prints colored playing courts on the screen with different colored backgrounds as well as movable playing pieces. The alphanumerics on the bottom of the screen look good too. The characters are quite high and legible and because it's done in graphics, you could even display Japanese or hieroglyphics.

Let's face it – it's a full-colored intelligent graphics display for only $180. This could be the last of the big-time bargains!! Judging from the component count as you can from the picture, I highly doubt Fairchild will be able to drop the price of this video game very much over the next few months, so now is probably as good a time as any to buy one and start working on it. Think of the possibilities of producing your own video games. Dissecting one of the video game cartridges showed that all it contained was two PSUs and these obviously had the four games that were included in that cartridge. By interfacing some EPROMs or RAMs through cartridge contacts, you would be able to run your own video games. Think of it – Lunar Landers with spaceships, Tank with several simultaneous playing pieces, and so on. There have been a good many articles written on graphic display and how to handle the mechanics of building a fixed background with movable objects. Possibly, with some dubious experimenting on your part, you'll be able to come up with a "Fairops" or "Gameops" brand new video game operating system written for the Fairchild F8 video game.

At the time of this writing, the schematic diagram of this game was not available. But by the time you read this article it's very likely that a complete schematic of the video game will be available, and then you'll be able to analyze what's going on in and around the microprocessor controlled game with very little difficulty. Also, Fairchild's release of full data on interconnecting F8 video game to the 8080 bus at some point in time in the near future will be invaluable to the serious computerist.

If you're interested in this and want more information, Mike Williams at Fairchild Xetron (3105 Alfred Street, Santa Clara CA 95050) is the man to contact. By the time this goes to press they should have those 8080-interface schematics finished . . . and available. ■



# Get a Watchdog

## *to monitor those real-time operations*

*Dave Brickner*
*205 E. Caribbean*
*Phoenix AZ 85022*

Applications of microprocessors in real-time situations such as process control are fascinating. The designer and builder have the satisfaction of watching great mechanical and chemical monsters knuckle under and dance to the tune of the robot powers of the computer and its program. Airplanes fly straight and level, trains always take the correct track, the house temperature is maintained at a precise 68°, and the burglar alarm separates family members from crooks. All of these actions happen precisely as painstakingly designed and programmed.

### What Happens if Something Goes Wrong?

But what if one instruction is incorrectly picked out of memory? What if the data suddenly reaches an overflow condition not anticipated in the design? These things can happen, and Murphy predicts they will! This sort of random fault in a real-time control system can have disastrous results. Enter Watchdog—a simple hardware circuit that will monitor real-time software and catch a large percentage of potential errors before disaster strikes.

Before we jump into the circuit, it is useful for us to recall some guidelines for the use of monitoring functions and relate these guidelines to the real-time situation.

### Monitor Functions

A few rules concerning monitor functions should be noted:

1. *Keep it simple.* I have seen system designers get so worried about all the things that could go wrong that the monitor system became the overriding factor. In this case, the system usually is never completed.
2. *If you can't help it, forget it.* For instance, if the power supply fails, you ain't gonna compute no more; so why try? (Of course, you can install two power supplies, but that returns us to rule one.)
3. *Only use monitors that catch lots of problems.* The corollary is: Don't use monitors that don't catch anything.
4. *When in doubt, do the least risky thing.*

Well, all that sounds obvious, but you'd better believe organizations like NASA have whole squads of people researching monitors and fault-detection schemes for their control systems. Most important for the hobbyist is to get the system up *with or without a monitor,* so rule one applies. Nobody likes to invest and get no return, so rule three applies. And, since we can't possibly think of everything, rules two and four apply.

### The Software

Now let's look at the real-time control situation to see the logic behind Watchdog. Fig. 1 shows a block diagram of a feedback-type real-time control system. The microprocessor measures the process output through its own input circuits and provides control of the process through its output circuit.

Fig. 2 is a simple view of the program for such a system. The predominant features are

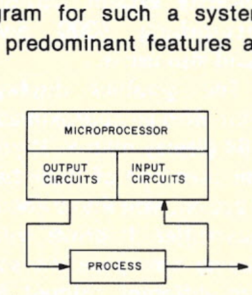



*Fig. 1. Feedback system for real-time control.*

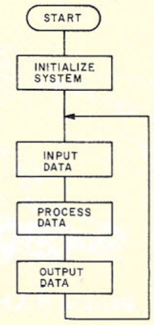



*Fig. 2. Real-time control system program.*

a program entry from start that initializes the system and a repeating program that recycles over and over through the system, measuring data and calculating adjustments to the output. Virtually every program in real time uses a variation of this scheme. Sure, it's possible to complicate the scheme with multiple branches in the repeated portion of the program or with multiple processes controlled in major and minor program cycles set by the interrupt. Our goal here is to look at the big picture—the overview that will help us select a cheap, useful monitor.

The repeating part of the program seems to be the area most useful to monitor since it is where most of the time is spent. The most obvious feature is the repeating nature of this portion of the program.

We expect the program to pass through the reentry of this part of the program on a predictable repeat basis. Each time the program gets to this point, we are reasonably sure it made it through the rest of the program. Herein lies the essence of the real-time monitor. Each time we pass through this program reentry, we output a pulse to Watchdog, which expects this pulse on a periodic basis. If an extended period goes by and no pulse arrives, the monitor assumes a failure has occurred and takes appropriate action.

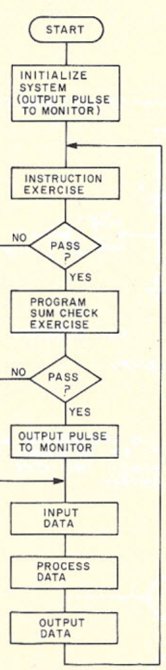



*Fig. 3. Instruction exerciser and sum check flowchart.*

### The Hardware

Perhaps you have realized that Watchdog is a simple, retriggerable, one-shot multivibrator. The 74123 style works just fine. One bit in one of the holding registers on an output port is connected to the one-shot input. On each pass through the program, the programmer sets and then resets this bit. During the initial design, the period of the one-shot is set longer than the longest period anticipated between the programmed pulse outputs. This period must be sufficiently long to prevent false alarms.

So far, you couldn't ask for a much simpler monitor. We have invested one discrete bit from an output port, one half of a 74123 microcircuit and two instructions. Any system fault causing the program to tight loop (continue to repeat a few instructions endlessly) or lose control will be caught by the Watchdog.

Further fault-detection capability can be added with a small additional software investment.

### Two Software Techniques

The two most effective software additions are an instruction exerciser and a sum check of memory. The instruction exerciser is a short subroutine that is executed with canned data and known results. The exercise should include several, if not all, of the most frequently used instructions. I usually execute this subroutine in the program just before sending the pulse to the one-shot. If the canned answer doesn't check, the output pulse is not sent and the monitor catches the error. In most programs it is sufficient to use an existing subroutine for the exercise.

The sum check is effective where program memory is contained in read-only memory or is at least completely static throughout the process. The program determines the arithmetic sum (ignoring overflow) of the entire read-only contents of memory periodically. This sum must be correct or no pulse goes out to the Watchdog. I typically set the program up so that on each pass one to ten bytes are added to the sum. This may take up to several seconds to get through a complete check, depending on the program length—but, better late than never. At least if a fault occurs I catch it after a few seconds, rather than not at all.

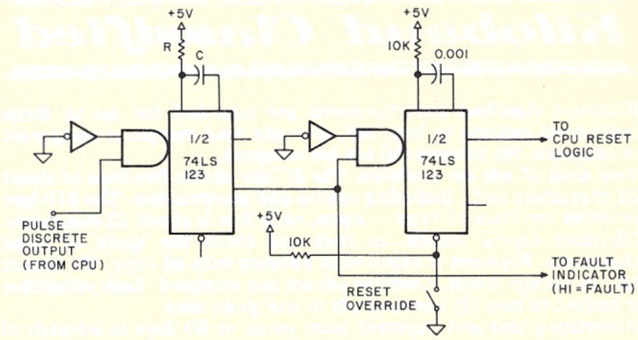



*Fig. 4. One-shot with override. Select R and C values so period is greater than longest anticipated in the process control program cycle.*

Fig. 3 is the new software flowchart. You may note that I have chosen to continue system operation even if the instruction exerciser or sum check fails. This is a matter of personal preference, especially concerning the process under control. You may decide this is too risky for the process you are controlling, in which case a simple shutdown would be more appropriate.

The logical question at this point is, how do we know the computer will follow our flowchart if it can't pass sum check or instruction exercise? The answer is faith in the idea that the more faulty our system becomes, the less likely it will pulse the Watchdog correctly.

### Implementing the Idea

So, there you have it; a few electronic parts and a dozen or so instructions and we have a monitor that will catch a large percentage of gremlins that may invade our microprocessor control system.

The output of the monitor can be used in a variety of ways. During debugging, or when human intervention is close by, I usually connect the Watchdog to an indicator light or audio alarm. If, however, the process is critical, fast or untouched by human hands, I connect the Watchdog into the CPU reset logic. This will cause the system to reset and pass through the initialization program. The reasoning is that the initialization program is designed to bring the system to a known quiet (stopped or idle) state. Thus, there is little likelihood of the process going completely astray, and a restart may sometimes clear the fault out of the system. Again, this is a design parameter each designer must select based on the requirements of the process under control.

One note: If you do elect to restart the system automatically from the monitor output, you must remember to override the monitor during program debugging or whenever the CPU is used for other tasks. Fig. 4 is a schematic of a 74123 dual retriggerable one-shot designed to provide Watchdog and the automatic reset with override.

I have used this type of monitor successfully in many aerospace applications and in my home designs. It satisfies all the rules of good monitor design and is simple and effective enough to provide much peace of mind for a very small investment.■

# Kilobaud Classified

Kilobaud classified advertisements are intended for use by those individuals desiring to buy, sell, or trade used computer equipment or software. No commercial ads are accepted.

Two sizes of ads are available. The $5 box allows five lines of about 22 characters each, including spaces and punctuation. The $10 box provides ten lines of type – again, each line is about 22 characters. Minimize capital letters, as they use twice the space of small characters. Payment is required in advance with ad copy. We cannot bill, or accept credit. Oversize ads are not accepted. Each subscriber is limited to two (2) identical ads in any given issue.

Advertising text and payment must reach us 60 days in advance of publication. For example, advertising copy for the March issue (mailed in February) must be in our hands on January 1. The publisher reserves the right to refuse a questionable or not applicable advertisement. Mail advertisements to: KILOBAUD CLASSIFIED, Kilobaud, Peterborough, NH 03458. Do not include any other material with your ad, as it may be delayed.

---

For Sale: Booklet—"How to Make Your Computer Pay (for itself)." Ideas and thoughts on ways to make and save money with your computer system. $5.50. Herbert Schildt, 1007 N. Division, Urbana IL 61801.

---

Mosley X-Y Plotter Model 2. Less than 5 hours use since last calibration. $110. J. D. Schoepf, Rt. 5 Box 113, St. Charles MO 63301.

---

ADM-3A with upper/lowercase. Brand new. Guaranteed working perfectly. $745. Write: Terminal, Box 2467, North Canton OH 44720.

---

PET users need info? Getting started with your PET workbook, $4. Workbook has helpful info, sample programs, exercises and fundamental features of PET BASIC. Send SASE for details. D. Smith, Box 921, Los Alamos NM 87544.

---

For Sale: 2 16K dynamic memory boards for S-100 bus. Assembled & tested. $300 each. Call Erick (415) 771-5496.

---

For Sale: Altair 8800B w/ broken protect switch (bat handle was accidently snapped off). Worked fine before the mishap. $650.

---

PET-2001 and Radio Shack TRS-80 arrived on campus. I want to survey users and report results to any interested hobbyists. Write: Professor Bill Parks, Walters State Community College, Morristown TN 37814.

---

For Sale: Iomec disk drive w/ single remov. 1 MByte platter w/supply, rack, manual. Also: CDC 300 1pm drum printer w/man. No interfaces. Sold "as is": neither used 3+ years. $1000 each. Chuck Gahan, 12781 Taylor St., Garden Grove CA 92645.

---

TRS-80 Monthly Summarizer PGM. HK-Level I Basic. Video and tape. Monthly summary by expense item. 26 Expense categories—you select. Great for monthly budget and year-end tax work. Documentative. Sat. guar. Tape $9.95. E. W. Barnes, 119 Skyline Dr., Plano TX 75074.

---

Pet Owners: Test your system before trying Las Vegas. 4 casino games on 1 cassette. Blackjack, craps, roulette, baccarat. Full lv rules and action graphics for hrs of fun. Spec. 4K or 8K. All 4 for $25. Guaranteed. CMS, 431 Monte Vista, Dallas TX 75223.

---

KIM Expansion-4K memory; KP-40 printer; power; keyboard. Assembled and working. Just plug in your KIM and go. I/O routines are included. $400 for all works. Beringer, 2500 Teddy, #29, Las Vegas NV 89102.

---

For sale: Teletype Model 33ASR, exc. cond., $600. Altair 8080A 8K, TTY and cassette I/O, perfect, $580. Selectric I/O typewriter with paper tape rdr/pch, $480 . . . all for $1500. A. Frankford, Lancaster PA (717) 299-2456.

---

For Sale: North Star BASIC programs: correspondence editor, $5; stock-market-analysis package; $5; mailing-list/random-access pack., $3; Spacewar game pack., $3; plus stk mkt data on disk. Includes P/E, price, volume, % yield-weekly averages for 1977 on 30 heavily traded corporations; only $25 for all 30! Send $5.25 or blank disk, write for complete list. Herbert Schildt, 1007 N. Division, Urbana IL 61801.

---

Conductive Foam. ¼" thick, 3¢/sq. in. plus 25¢ postage. WF, 713 George Ln., Glendora NJ 08029.

---

CUTS fix: Unacceptable error rate with otherwise functional board is probably not your fault! Send $2 for copyrighted doc. of board mods. and explanation of the problem. DIAS, 234 Union St., Schenectady NY 12305.

---

**Use Kilobaud Classified**

# Contest!

The year 1977, at least for purposes of this contest, will end in September—since the first contest results appeared in the October 1977 issue. Winner for the year, then, should be announced in late 1978.

Meanwhile, the winner for the best article in the January 1978 issue is Ed Juge, author of "The TRS-80: how does it stack up?"

The book winner this month is Robert C. Boyd of Kennebunkport ME.

Keep those votes coming!

# Updates

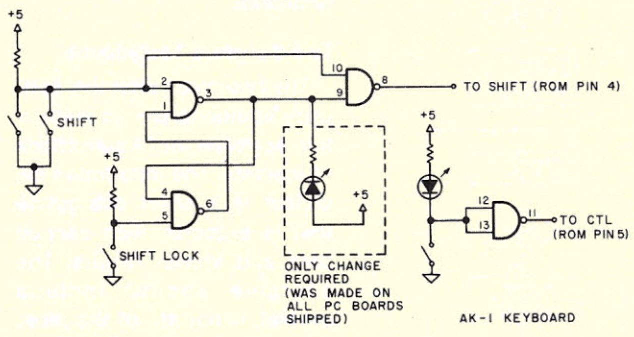
So many people have called requesting the phone number of Larry McCaig (author of "Small Business Software," Parts 1 and 2, *Kilobaud* Nos. 14 and 15) that we are going to print it. That number is (207) 487-2219.

The KB Club Calendar, which has been absent for two issues, will appear again in the next issue—barring any more blizzards, a factor in holding up compilation of the Club Calendar.

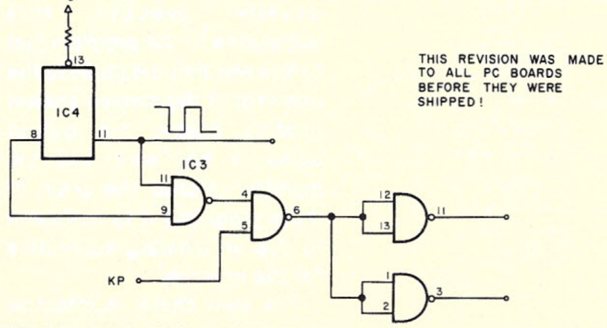
# CORRECTIONS

These circuit revisions to "Build Your Own ASCII Keyboard" by Robert Brehm (*Kilobaud* No 9, page 22) were sent to us by Bob.

*REVISED SHIFT CIRCUIT*




*REVISED REPEAT CIRCUIT - AKI KEYBOARD*



# KB CALENDAR

### Atlanta GA

Papers are invited for presentation at the 16th Annual Convention of the Association for Educational Data Systems, Atlanta GA, May 15-19, 1978. For further information, contact: Dr. James E. Eisele, Office of Computing Activities, University of Georgia, Athens GA 30602.

### Blacksburg VA

Tychon, Incorporated, announces the start of their 1978 microcomputer course programs in April at their learning center in Blacksburg. Three microcomputer courses will be offered: No. 628, Microprocessor Interfacing, April 6-8; No. 685, Introduction to Assembly Language Programming for 8080/8085 Processors, April 10-12; No. 687, Intermediate Assembly Language Programming for 8080/8085 Processors, April 13-15. Each course is three days long and the cost is $295 per person, per course. For more information, please call Dr. Chris Titus, course director, at (703) 951-9030.

### Washington DC

Amateur Computing 78 microcomputer festival will be held July 22-23 at the Sheraton National Motor Hotel, Arlington VA. Those interested in presenting a paper, participating in a panel discussion, displaying an amateur computer system or sponsoring a tutorial should submit a letter of intent along with a one-page abstract or outline by April 15 to John Wall Miller, Program Chairman, 6921 Pacific Lane, Annandale VA 22003, (703) 256-5702. Authors will be provided with instructions for preparation of camera-ready papers, which are due by June 1.

For information, write AMRAD, PO Box 682, McLean VA 22101.

### Long Beach CA

PERCOMP '78 (co-sponsored by the International Computer Society/SCCS and the Rockwell Hobbyist Computer Club) will be held at the Long Beach Convention Center, Long Beach CA, April 28-30, 1978. PERCOMP is a selling show designed with the home computerist and small-business person in mind. For information concerning seminars, contact James Lindwedel, Technical Program Chairperson, PERCOMP '78, 1833 E. 17th St., Santa Ana CA 92701.

NLS

# NLS LEADS AGAIN!!
## THE NEW MS-15 MINISCOPE

PORTABLE
BATTERY OPERATED
15 MEGAHERTZ
**OSCILLOSCOPE**


### FEATURES

- **15 megahertz bandwidth.**
- **External and internal trigger.**
- **Time Base — 0.1 microseconds to 0.5 Sec/div — 21 settings.**
- **Battery or line operation.**
- **Automatic and line sync modes.**
- **Power consumption less than 15W.**
- **Vertical Gain — 0.01 to 50 volts/div — 12 settings.**
- **Weight is only 3 pounds.**

*From the originator of the Digital Voltmeter, Non-Linear Systems comes the MS-15 Miniscope. It is a fine electronic instrument with a great deal of measuring capability and excellent accuracy. Its design is modern, utilizing the latest in low-powered integrated circuits, and it is packaged into the smallest practical size. The instrument fits into many briefcases and tool boxes with room to spare.*

*Operating characteristics have been chosen so that the MS-15 will make all of the measurements needed in servicing most electronic equipment. It is field-portable so its use is not restricted to the bench.*

## SPECIFICATIONS:

**Y Axis**

| | |
|---|---|
| **Vertical Input:** | x1-10mV/div to 10V/div in four ranges, each continuously variable. x2-20mV/div to 20V/div in four ranges, each continuously variable. x5-50mV/div to 50V/div in four ranges, each continuously variable. Accuracy is 3% |
| **Input Impedance:** | 1M ohm shunted by 50 pF |
| **Input Voltage:** | 350 volts peak |
| **Bandwidth (3 db points):** | DC/DC to 15 MHz; AC 5 Hz to 15 MHz |
| **Internal Calibrator:** | 1 .0V (±5%) peak-to-peak square wave at 10-20 kHz |
| **Graticule:** | 4 div Y by 5 div X; 1 div = 0.25 inch. Viewing area: 1.1''H x 1.35''W. |

**X Axis**

| | |
|---|---|
| **Horizontal Input Bandwidth (3 db points):** | DC to 200 kHz |
| **Input Impedance:** | 1M ohm shunted by 50 pF |
| **Sensitivity:** | 1V/div, 100V peak maximum input. |
| **Time Base:** | x1, uS-0.1uS/div to 100 uS/div. x 2, uS-0.2uS/div to 200 uS/div. x5, uS-0.5uS/div to 500 uS/div. x1, mS-0.1mS/div to 100 mS/div. x2, mS-0.2mS/div to 200 mS/div. x5, mS-0.5mS/div to 500 mS/div. all in four ranges, each continuously variable. (Range increments are .1, 1, 10, 100.) With vernier in full clockwise position, calibrated time measurements are possible. Accuracy is 3%. |

**Triggering**

| | |
|---|---|
| **Internal:** | Sweep triggered from internal trigger source. |
| **Automatic:** | Trigger source is internal calibrator frequency. To be used if there is no other trigger source available to synchronize the sweep. |
| **Line:** | Trigger is derived from line frequency when using the battery charger. |
| **Slope:** | Selects sync to positive- or negative-going waveform. |
| **Sensitivity:** | Less than one division. |
| **External:** | Controls function as for internal triggering. |

**Power**

| | |
|---|---|
| **Batteries:** | Three self-contained lead acid ''D'' cells. |
| **Operating Time:** | Typically 4 hours. |
| **Charging Time Scope Operating:** | Will run indefinitely but not reach full charge. |
| **Non-operating:** | Sixteen hours. |
| **External Power:** | Battery charger 115 vac (220 vac on request), 50-60Hz, less than 15 watts. |
| **Dimensions:** | 3.1''H x 6.4''W x 8.0''D. |
| **Weight:** | Three pounds. |
| **Accessories Standard:** | Tilt stand, battery charger, input cable and miniature banana plugs (4). |
| **Options:** | Leather carrying case and 10-1 attenuating probe, 10M ohm input. |
| **Warranty:** | One year parts and labor. |

### MS-15 with Rechargeable Batteries and Charger
### $289.00

### Leather Carrying Case

The leather case has 2 separate compartments. One to hold the scope, the other to hold the charger, probe, shoulder strap, etc. The case can be worn on the belt, or over the neck.

The snaps used on the case are "one way", thus accidental striking of the case against an object will not undo the snaps or let it be pulled off your belt.

**41-140** **$30.00**

### Probe

10 to 1 probe with 10 megohm input. Probe uses spring hook tip for sure connection. Compensation network is located at the connector rather than at the probe, so as to keep size and weight to a minimum.

**41-141** **$24.50**

## FREE PROBE

Just send or mention this COUPON and KILOBAUD MAGAZINE and receive 41-141 PROBE FREE with purchase of MS-15 MINISCOPE.

P21

**PRIORITY 1 ONE ELECTRONICS**

10031 Woodley Ave. Sepulveda CA. 91343

Terms: VISA, MC, BAC, check, Money Order, C.O.D., U.S. Funds Only. CA residents add 6% sales tax. Minimum order $10.00. Orders less than $75.00 include 10% shipping and handling, excess refunded. Just in case . . . please include your phone no.

**"Sorry, no over the counter sales" phone orders welcome (213) 893-8202**

**Good thru June 1978**

OEM and Institutional inquiries invited.

PRIORITY 1 ELECTRONICS a div. of theHEATH Group.

...chure.


NLS MS-15 MINISCOPE $289.00
NLS MS-15 MINISCOPE $289.00
Vector
8800V
Universal Microcomputer/processor plugboard, use with S-100 bus. Complete with heat sink & hardware. 5.3'' x 10'' x 1/16''
1-4 $19.95 5-9 $17.95 10-24 $15.96
8801-1
Same as 8800V except plain; less power buses & heat sink.
1-4 $14.95 5-9 $13.46 10-24 $11.96
3662 6.5'' x 4.5'' $7.65
3662-2 9.6'' x 4.5'' $11.45
P pattern plugboards for IC's Epoxy Glass 1/16'' 44 pin con. spaced .156
R644 P.C. receptacle 22/44 cont. .156 ctrs. pierced solder eyelets tails $4.49
R644-3 P.C. receptacle 22/44 cont. .156 ctrs Wire Wrap tails gold $4.49
Bus P.C. Edge Connectors
R681 for ALTAIR mother boards 100 contacts (2 rows of 50) .125 spaced
R681-2 .025 sq. wire wrap posts $7.35
R681-3 pierced solder eyelet tails $7.35
8803 MOTHER BOARD FOR S100 BUS MICRO-COMPUTERS
Price: $29.50
NEW VOLKSMETERS!
With LCD Display—Excellent Readability in Direct Sunlight!
LM-300 Full three digits $99.50*
LM-350 3½ digits $125.00*
FEATURES
• Measures DC volts, AC volts, ohms and current.
• Automatic polarity, decimal and overload indication.
• No zero adjustment and no full scale ohms adjust.
• Large LCD display for easy reading without interpolation.
• Size: 1.9'' H x 2.7'' W x 4.0'' D.
• Parts and labor guaranteed for one year.
Standard AA-size batteries provide up to 20 hours of operation. Rechargeable NiCad batteries and charger unit available as optional equipment.
*Batteries not included.
3690-12
Card Extender has 100 contacts — 50 per side on .125 centers — Attached connector — is compatible with S-100 Bus Systems $25.00
3690 6.5'' 22/44 pin .156 ctrs. Extenders $12.00
.8'' LED ALARM CLOCK
12 hr. LED Alarm Clock uses 3½ digit .8'' LED Display with AM/PM indicators and colons. Direct drive, PIN to PIN interface with S1998A I.C. Just add switches, AC Supply, Alarm. Display and I.C. only.
$7.95 or 2/$15.00
LIQUID CRYSTAL DIGITAL CLOCK-CALENDAR
SPECIAL
• For Auto, Home, Office
• Small in size
• Push button for seconds release for date.
$33.95
WRAP POST
SLIT-N-WRAP
Wraps insulated wire on .025'' square posts
FOUR TIMES FASTER
NO PRE-STRIPPING* NO PRE-CUTTING* SPOOL-FED WIRE*
P180 with two 100' spools of 28 ga. wire $24.95
P160-4T charger wire $75.00
SLIT-N-WRAP WIRE NO. 28 GAGE INSULATED WIRE, 100' SPOOLS
$2.75 Package
DIRECT POSITIVE PHOTO-RESIST Etched Circuit Kit
P1-VPK $7.95 or 2/15.00
14 & 16 PIN 3 LEVEL WIRE WRAP SOCKETS
14-T3 100 for $30.00
16-T3 100 for $30.00
50 of ea. for $32.00
MICRO-KLIP
for .042 dia. holes
SC-5 With Rechargeable Batteries & Charger Unit $79
FM-7 With Rechargeable Batteries & Charger Unit $168
SALE
Joypot 100 K $4.95
2708 8K 450 ns EPROM FACTORY PRIME
1-9 $13.00
10-24 $12.50
25+ Call for Price
Circuit Evaluator for fast, solderless, plug-in circuit building and testing
ACE Model 236
1/16 Vector BOARD
.042 dia holes on 0.1 spacing for IC's
3 LEVEL GOLD WIRE WRAP SOCKETS
2102L1 450ns Low Power—High Speed RAM
8-24 $1.50
25-99 $1.40
100-249 $1.25
250+ $1.19
Price Breakthrough: MA1003 CAR CLOCK
Bright Green Fluorescent Display Crystal Time Base Assembled, just add switches and 12 VDC
$17.95
Free
Just send or mention this COUPON and MAGAZINE and receive with your order — DIP PROJECT BOARDS — assorted sizes and types, epoxy glass, prepunched and copperclad Boards.
3682 9.6'' x 4.5'' $10.97
3682-2 6.5'' x 4.5'' $9.81
Hi-Density Dual-In-Line Plugboard for Wire Wrap with Power & Grd. Bus Epoxy Glass 1/16'' 44 pin con. spaced .156
3677 9.6'' x 4.5'' $10.90
3677-2 6.5'' x 4.5'' $9.74
Gen. Purpose D.I.P. Boards with Bus Pattern for Solder or Wire Wrap. Epoxy Glass 1/16'' 44 pin con. spaced .156
SPECIAL
LEDU MG 10A List $72.00
PRIORITY ONE ELECTRONICS
10031C Woodley Ave. Sepulveda CA. 91343
P21
Terms: VISA, MC, BAC, check, Money Order, C.O.D., U.S. Funds Only. CA residents add 6% sales tax. Minimum order $10.00. Orders less than $75.00 include 10% shipping and handling; excess refunded. Just in case . . . please include your phone no.
Good thru June 1978
"Sorry, no over the counter sales"
phone orders welcome (213) 893-8202
Send for our latest brochure.
OEM and Institutional inquiries invited.

# THE CHIPS ARE DOWN!

## 8K NOW JUST $149 ASSEMBLED

Thinker Toys™ brings down the high cost of adding big memory capacity to your S-100 system!

The ECONORAM III* 8Kx8 (by Morrow's Micro-Stuff) comes fully assembled, burned in, tested and fully warranted for one full year—*for just $149!*

It's configured as two individually addressable 4K blocks. And it typically consumes less than one-half the power of any competitively-priced memory.

Obviously, our new ECONORAM III* 8Kx8 isn't just another cheap kit. It's a design breakthrough in dynamic memory that gives you guaranteed reliability with tremendous savings in cost and power.

You save on cost and power because ECONORAM III* is the first fully reliable dynamic memory. It's unique SynchroFresh™ refresh circuitry weaves itself into the natural timing of the S-100 bus to provide simple, invisible and absolutely reliable refresh . . . with fewer parts, less power, lower cost and no interference with computer speed or timing.

You just plug ECONORAM III* into your S-100 bus and run perfectly for at least one full year. Or your money fully refunded. It's that simple.

So don't let the high cost of memory keep your system down. ECONORAM III* 8Kx8 memory is available now at leading computer shops. Or ask your nearest computer shop to order it for you.

Or, if unavailable locally, write direct to Thinker Toys™ for specs and ordering. BAC/MC orders can be placed by phone to 415-527-7548 (10-4 PST). Add $3 for handling; Cal. res. add tax.

**Morrow's Micro-Stuff for**

# Thinker Toys™

George Morrow-Inventor & Designer

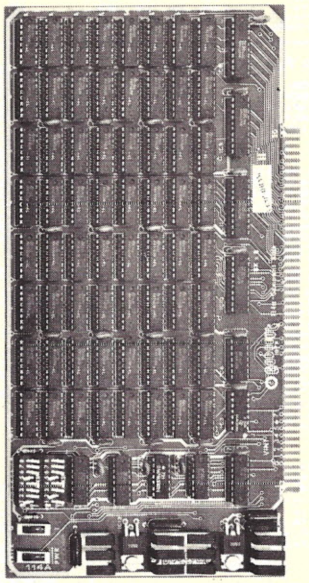
**1201 10th Street Berkeley, CA 94710**

***ECONORAM is a trademark of Godbout Electronics**

M7

# COMPUTER CORNUCOPIA:

## 8K x 8 Econoram II™



**Kit $135**
**Assembled $155**
**3 kits for $375**

**This is the board that thousands of owners swear on, not at.** There are lots of reasons, such as unique addressing options, reliability, full buffering, static operation, fast access time, a full set of sockets . . . but probably the most popular feature is the price, which is all the more remarkable because of the high level of quality. One owner reviewed this board in the 1/77 issue of **Kilobaud**, closing with the words "If you're not convinced by now that the Econoram II is one of the best memory buys on the market today, you really have to be one tough cookie—either that or you work for someone else who makes memory boards".

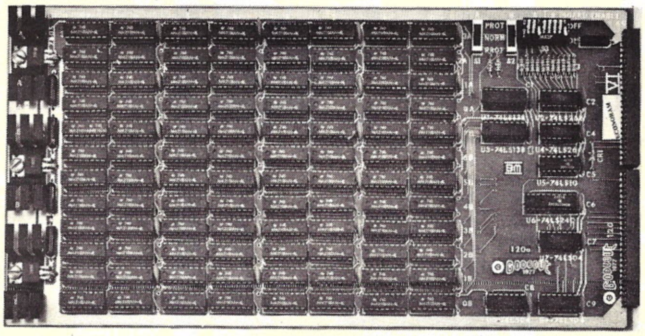
## MESSAGE:

Now you can have the quality of Godbout products and the convenience of local shopping . . . many computer stores and other electronic outlets stock a number of our items, from individual components to complete board kits.

We always welcome new dealers; if you have a store and want to know more about adding these popular products to your line, write or phone and ask for our dealer package.

Finally, we'd like to add that even inveterate optimists such as ourselves have been amazed and delighted at the response to our computer kits. We thank you very much for your support (and your patience during our period of heaviest expansion); it will enable us to keep you excited with more new goodies in the months ahead.

## 12K Econoram VI™: $235 (Kit)



**We proudly welcome our newest memory board family member, designed from the ground up for full compatibility with the Heath Company H8.** Organized as two independent blocks, one 8K and one 4K. Has the same basic features as our Econoram II™—all static design, switch selected protect and phantom, sockets for all ICs, full buffering on address and data lines—plus the required hardware and edge connector to mate mechanically with the H8. You can have our 12K board for the price of the Heath Company's 8K . . . with the performance you have come to expect from products carrying the Econoram trade mark.

## OTHER POPULAR ITEMS

**"8K X 8 ECONORAM III™"** Dynamic memories have a reputation for low power, economy — and difficulty of use. If you'd like to have the good without the bad, then this is the board for you. Not a kit; comes assembled, tested, and ready to run in any S-100 machine (Altair, IMSAI, etc.). Configuration as two separate 4K blocks. Zero wait states with 8080 CPU. 1 year warranty. **$149.00**

**"Vector #8803 Bare 10 Slot Motherboard"** This is a version of our 10 slot motherboard, but contains no edge connectors, active termination circuitry, or other components — just the board. Great for experimenters or people with small systems who don't need the full kit. **$29.50**

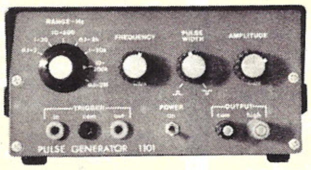
**"10 Slot Motherboard".** Use with the IMSAI microcomputer as an add-on with room for 10 peripherals, or for starting an 11 slot standalone system. Comes with all edge connectors, and includes an on-board active termination circuit to minimize the crosstalk, noise, overshoot, and ringing that can occur with unterminated boards. Epoxy glass, solder masked board, with bypass caps and heavy power traces. **#CK-015, $90.00.** Kit form.

**"18 Slot Motherboard".** Same as the 10 Slot Motherboard except with 18 slots. **#CK-016, $124.00.** Kit form.

**"12V 8A Power Supply".** Handles 12A peaks with 50% duty cycle. Includes crowbar overvoltage protection, current limiting, adjustable output 11-14V, custom wound transformer. Easy assembly: all parts except transformer/filter cap/diodes mount on circuit board, including heat sinks and power transistors. While designed for powering automotive equipment in the home, many users report success using these for powering some disk drive systems. **#HK-104, $44.50.** Kit form. Please include sufficient postage.

**DB-25 RS-232 SUBMINI-D CONNECTORS** Male plug, **#CK-1004, $3.95;** female jack, **#CK-1005, $3.95;** plastic hood for male connector, **#CK-1006, $0.90.**

**"Active Terminator Board".** For those of you who have a motherboard without active terminations, plug in this card and obtain the benefits of improved signal transfer, thanks to less noise, crosstalk, ringing, and overshoot. Same circuitry as used in our motherboards. **#CK-017, $29.50.** Kit form.




**"Interdesign Model 1101 Pulse Generator".** Ideal for clocking TTL circuits, from .1 Hz to 2 MHz. 20 to 1 frequency spread for each band. 20% to 80% duty cycle minimum, fully triggerable. Portable for field use, and includes rechargeable nicads and charger for either battery charge or AC operation. If you don't have a pulse generator, here's one at a good price. **#Z-006, $90.00.** Fully assembled and tested.

We got our start in this business selling components to fanatical hobbyists who couldn't find the parts of their dreams anywhere else. What with all the attention being lavished on our computer kits, music kits, and hobby kits, we wanted to take this opportunity to remind you that we still sell parts . . . at the same outrageously low prices that got us going in the first place.

Need some resistors? Capacitors? Or maybe some ICs . . . CMOS . . . linears . . . memories . . . Vector products (like the Slit-N-Wrap or wire pencil) . . . or a really classy chassis to dress up your next project . . . we've got all that, and then some.

How about an American made, **Switchcraft RCA phono jack?** (Closed circuit too — that's a rare bird). Comes mounted on a little plastic carrier with screw holes for easy mounting. At **7 for $1**, here is a chance to stock up on some top quality connectors.

## 74 LS TTL

| | |
|---|---|
| 74LS00 | 0.30 |
| 74LS01 | 0.30 |
| 74LS02 | 0.30 |
| 74LS04 | 0.33 |
| 74LS08 | 0.36 |
| 74LS10 | 0.30 |
| 74LS11 | 0.36 |
| 74LS12 | 0.33 |
| 74LS14 | 1.38 |
| 74LS15 | 0.30 |
| 74LS20 | 0.30 |
| 74LS21 | 0.33 |
| 74LS22 | 0.33 |
| 74LS26 | 0.43 |
| 74LS27 | 0.36 |
| 74LS30 | 0.30 |
| 74LS32 | 0.38 |
| 74LS37 | 0.45 |
| 74LS38 | 0.45 |
| 74LS42 | 0.98 |
| 74LS47 | 1.00 |
| 74LS48 | 0.98 |
| 74LS74 | 0.50 |
| 74LS75 | 0.68 |
| 74LS76 | 0.50 |
| 74LS86 | 0.50 |
| 74LS109 | 0.50 |
| 74LS125 | 0.63 |
| 74LS126 | 0.63 |
| 74LS132 | 1.25 |
| 74LS138 | 1.10 |
| 74LS139 | 1.15 |
| 74LS151 | 0.95 |
| 74LS155 | 1.38 |
| 74LS157 | 0.95 |
| 74LS160 | 1.40 |
| 74LS161 | 1.40 |
| 74LS162 | 1.40 |
| 74LS163 | 1.40 |
| 74LS168 | 1.87 |
| 74LS169 | 1.87 |
| 74LS173 | 1.65 |
| 74LS174 | 1.25 |
| 74LS175 | 1.15 |
| 74LS240 | 1.88 |
| 74LS257 | 1.25 |
| 74LS258 | 1.25 |
| 74LS266 | 0.53 |
| 74LS283 | 1.20 |
| 74LS365/80LS95 | 0.75 |
| 74LS366/80LS96 | 0.75 |
| 74LS367/80LS97 | 0.75 |
| 74LS368/80LS98 | 0.75 |
| 74LS386 | 0.55 |
| 74LS95 | 1.13 |
| 74LS96 | 1.13 |
| 74LS97 | 1.13 |
| 74LS98 | 1.13 |

send for our flyer!

TERMS: Please allow up to 5% for shipping; excess refunded. Californians add tax. COD orders accepted with street address for UPS. For VISA® /Mastercharge® orders call our 24 hour order desk at **(415) 562-0636.** Prices good through cover month of magazine.

GODBOUT
BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

FREE FLYER: These are just a few of the items we carry for the computer enthusiast. We also stock a broad line of semiconductors, passive components, and hobbyist items. We will gladly send you a flyer describing our products upon receipt of your name and address.

G4

# 7400N TTL

| | | | | | |
|---|---|---|---|---|---|
| SN7400N | .16 | SN7472N | .39 | SN74160N | 1.25 |
| SN7401N | .18 | SN7473N | .39 | SN74161N | .99 |
| SN7402N | .20 | SN7474N | .35 | SN74162N | 1.95 |
| SN7403N | .20 | SN7475N | .50 | SN74163N | .99 |
| SN7404N | .20 | SN7476N | .35 | SN74164N | .99 |
| SN7405N | .20 | SN7479N | 5.00 | SN74165N | .99 |
| SN7406N | .20 | SN7480N | .50 | SN74166N | 1.25 |
| SN7407N | .35 | SN7482N | .99 | SN74167N | 3.25 |
| SN7408N | .20 | SN7483N | .70 | SN74170N | 2.10 |
| SN7409N | .25 | SN7485N | .89 | SN74172N | 6.00 |
| SN7410N | .20 | SN7486N | .39 | SN74173N | 1.50 |
| SN7411N | .30 | SN7488N | 3.50 | SN74174N | 1.25 |
| SN7412N | .35 | SN7489N | 2.49 | SN74175N | .99 |
| SN7413N | .69 | SN7490N | .45 | SN74176N | .79 |
| SN7414N | .70 | SN7491N | .75 | SN74177N | .79 |
| SN7416N | .35 | SN7492N | .49 | SN74179N | 2.49 |
| SN7417N | .35 | SN7493N | .49 | SN74180N | .99 |
| SN7420N | .20 | SN7494N | .79 | SN74181N | 2.49 |
| SN7421N | .39 | SN7495N | .79 | SN74182N | .95 |
| SN7422N | .49 | SN7496N | .79 | SN74184N | 1.95 |
| SN7423N | .37 | SN7497N | 3.00 | SN74185N | 1.95 |
| SN7425N | .29 | SN74100N | 1.25 | SN74186N | 15.00 |
| SN7426N | .29 | SN74107N | .39 | SN74187N | 6.00 |
| SN7427N | .37 | SN74109N | .95 | SN74188N | 3.95 |
| SN7429N | .42 | SN74116N | 1.95 | SN74190N | 1.79 |
| SN7430N | .25 | SN74121N | .39 | SN74191N | 1.25 |
| SN7432N | .25 | SN74122N | .39 | SN74192N | .89 |
| SN7437N | .35 | SN74123N | .50 | SN74193N | .89 |
| SN7438N | .35 | SN74125N | .60 | SN74194N | 1.25 |
| SN7439N | .25 | SN74126N | .60 | SN74195N | .75 |
| SN7440N | .21 | SN74132N | 1.25 | SN74196N | 1.00 |
| SN7441N | .89 | SN74136N | .95 | SN74197N | 1.00 |
| SN7442N | .69 | SN74141N | 1.15 | SN74198N | 1.75 |
| SN7443N | .75 | SN74142N | 2.95 | SN74199N | 1.75 |
| SN7444N | .75 | SN74143N | 3.25 | SN74200N | 5.59 |
| SN7445N | .75 | SN74144N | 3.00 | SN74251N | 1.79 |
| SN7446N | .89 | SN74145N | 1.15 | SN74279N | .90 |
| SN7447N | .69 | SN74147N | 2.35 | SN74283N | 2.25 |
| SN7448N | .89 | SN74148N | 2.00 | SN74284N | 6.00 |
| SN7450N | .25 | SN74150N | 1.25 | SN74285N | 6.00 |
| SN7451N | .25 | SN74151N | .79 | SN74365N | .75 |
| SN7453N | .25 | SN74153N | .89 | SN74366N | .75 |
| SN7454N | .25 | SN74154N | 1.25 | SN74367N | .75 |
| SN7459A | .25 | SN74155N | .89 | SN74368N | .75 |
| SN7460N | .25 | SN74156N | .89 | SN74390N | 2.25 |
| SN7470N | .45 | SN74157N | .89 | SN74393N | 2.25 |

20% Discount for 100 Combined 7400's

# CMOS

| | | | | | |
|---|---|---|---|---|---|
| CD4000 | .23 | CD4044 | .89 | CD4520 | 1.29 |
| CD4001 | .23 | CD4046 | 1.79 | MC14562 | 14.50 |
| CD4002 | .23 | CD4047 | 2.50 | CD4566 | 2.25 |
| CD4006 | 1.19 | CD4048 | 1.35 | MC14583 | 3.50 |
| CD4007 | .25 | CD4049 | .49 | 74C00 Series | |
| CD4009 | .49 | CD4050 | .49 | 74C00 | .39 |
| CD4010 | .49 | CD4051 | 1.19 | 74C02 | .55 |
| CD4011 | .23 | CD4053 | 1.19 | 74C04 | .75 |
| CD4012 | .25 | CD4056 | 1.49 | 74C08 | 1.00 |
| CD4013 | .39 | CD4059 | 9.95 | 74C10 | .65 |
| CD4014 | 1.39 | CD4060 | 1.49 | 74C14 | 3.00 |
| CD4015 | 1.19 | CD4066 | .79 | 74C20 | .65 |
| CD4016 | .49 | CD4068 | .39 | 74C30 | .65 |
| CD4017 | 1.19 | CD4069 | .45 | 74C42 | 2.15 |
| CD4018 | .99 | CD4070 | .55 | 74C73 | 1.50 |
| CD4019 | .49 | CD4071 | .23 | 74C74 | 1.15 |
| CD4020 | 1.19 | CD4072 | .49 | 74C89 | 4.00 |
| CD4021 | 1.39 | CD4076 | 1.39 | 74C90 | 3.00 |
| CD4022 | 1.19 | CD4081 | .23 | 74C93 | 2.00 |
| CD4023 | .23 | CD4082 | .23 | 74C95 | 2.00 |
| CD4024 | .79 | CD4098 | 2.49 | 74C107 | 1.25 |
| CD4025 | .23 | MC14409 | 14.95 | 74C151 | 2.90 |
| CD4026 | 2.25 | MC14410 | 14.95 | 74C154 | 3.00 |
| CD4027 | .69 | MC14411 | 14.95 | 74C157 | 2.15 |
| CD4028 | .89 | MC14419 | 4.95 | 74C160 | 3.25 |
| CD4029 | 1.19 | MC14506 | .75 | 74C161 | 3.25 |
| CD4030 | .49 | MC14507 | .99 | 74C163 | 3.00 |
| CD4035 | .99 | CD4508 | 3.95 | 74C164 | 3.25 |
| CD4040 | 1.19 | CD4510 | 1.39 | 74C173 | 2.60 |
| CD4041 | 1.25 | CD4511 | 1.29 | 74C193 | 2.75 |
| CD4042 | .99 | CD4515 | 2.95 | 74C195 | 2.75 |
| | | CD4518 | 1.29 | 80C95 | 1.50 |
| | | | | 80C97 | 1.50 |

# LINEAR

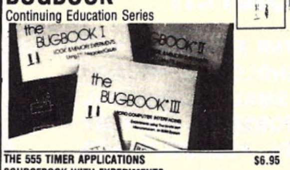
| | | | | | |
|---|---|---|---|---|---|
| LM300H | .80 | LM340T-8 | 1.25 | LM739N | 1.19 |
| LM301H | .35 | LM340T-12 | 1.25 | LM741CH | .35 |
| LM301CN | .35 | LM340T-15 | 1.25 | LM741CN | .35 |
| LM302H | .75 | LM340T-18 | 1.25 | LM741-14N | .39 |
| LM304H | 1.00 | LM340T-24 | 1.25 | LM747H | .79 |
| LM305H | .60 | LM350N | 1.00 | LM747N | .79 |
| LM307CN | .35 | LM351CN | .65 | LM748H | .39 |
| LM308H | 1.00 | 78MG | 1.75 | LM748N | .39 |
| LM308CN | 1.00 | LM370N | 1.15 | LM1303N | .90 |
| LM309H | 1.10 | LM373N | 3.25 | LM1304N | 1.19 |
| LM309K | 1.25 | LM377N | 4.00 | LM1305N | 1.40 |
| LM310CN | 1.15 | LM380N | 1.25 | LM1307N | .85 |
| LM311H | .90 | LM380CN | .99 | LM1310N | 2.95 |
| LM311N | .90 | LM381N | 1.79 | LM1351N | 1.65 |
| LM317K | 6.50 | LM382N | 1.79 | LM1414N | 1.75 |
| LM318CN | 1.50 | NE501K | 8.00 | LM1458CN | .59 |
| LM319N | 1.30 | NE510A | 6.00 | LM1496N | .95 |
| LM320K-5 | 1.35 | NE529A | 4.95 | LM1556V | 1.75 |
| LM320K-5.2 | 1.35 | NE531H | 3.00 | LM2111N | 1.95 |
| LM320K-12 | 1.35 | NE536T | 6.00 | LM2901N | 2.95 |
| LM320K-15 | 1.35 | NE540L | 6.00 | LM3053 | 1.50 |
| LM320T-5 | 1.25 | NE550N | 1.30 | LM3065N | .69 |
| LM320T-5.2 | 1.25 | NE555V | .39 | LM3900N(3401) | .49 |
| LM320T-8 | 1.25 | NE560B | 5.00 | LM3905N | .89 |
| LM320T-12 | 1.25 | NE561B | 5.00 | LM3909 | 1.25 |
| LM320T-15 | 1.25 | NE562B | 5.00 | LM5556N | 1.85 |
| LM320T-18 | 1.25 | NE565H | 1.75 | MC5558V | 1.00 |
| LM320T-24 | 1.25 | NE565N | 1.25 | LM7525N | .90 |
| LM323K-5 | 5.95 | NE566CN | 1.75 | LM7534N | .75 |
| LM324N | 1.80 | NE567H | 1.95 | 8038B | 4.95 |
| LM339N | .99 | NE567V | 1.49 | LM75450 | .49 |
| LM340K-5 | 1.35 | LM703CN | .45 | 75451CN | .39 |
| LM340K-6 | 1.35 | LM709H | .29 | 75452CN | .39 |
| LM340K-8 | 1.35 | LM709N | .29 | 75453CN | .39 |
| LM340K-12 | 1.35 | LM710N | .79 | 75454CN | .39 |
| LM340K-15 | 1.35 | LM711N | .39 | 75491CN | .79 |
| LM340K-18 | 1.35 | LM723H | .55 | 75492CN | .89 |
| LM340K-24 | 1.35 | LM723N | .55 | 75494CN | .89 |
| LM340T-5 | 1.25 | LM733N | 1.00 | RC4151 | 5.95 |
| LM340T-6 | 1.25 | | | RC4194 | 5.95 |
| | | | | RC4195 | 3.25 |

# 74LS00 TTL

| | | | | | |
|---|---|---|---|---|---|
| 74LS00 | .29 | 74LS76 | .49 | 74LS155 | 1.25 |
| 74LS02 | .29 | 74LS83 | 1.75 | 74LS157 | 1.50 |
| 74LS03 | .29 | 74LS85 | 2.49 | 74LS160 | 1.95 |
| 74LS04 | .35 | 74LS86 | .49 | 74LS161 | 1.95 |
| 74LS05 | .35 | 74LS90 | .89 | 74LS162 | 1.95 |
| 74LS08 | .29 | 74LS92 | .89 | 74LS163 | 1.95 |
| 74LS10 | .29 | 74LS93 | .89 | 74LS164 | 1.95 |
| 74LS13 | .69 | 74LS95 | 1.50 | 74LS175 | 1.95 |
| 74LS14 | 1.75 | 74LS96 | 1.89 | 74LS181 | 3.69 |
| 74LS20 | .29 | 74LS107 | .59 | 74LS190 | 2.49 |
| 74LS26 | .39 | 74LS109 | .59 | 74LS191 | 2.49 |
| 74LS27 | .39 | 74LS112 | .59 | 74LS192 | 2.49 |
| 74LS28 | .39 | 74LS113 | .59 | 74LS193 | 2.49 |
| 74LS30 | .29 | 74LS132 | 1.25 | 74LS194 | 1.89 |
| 74LS32 | .39 | 74LS136 | .59 | 74LS195 | 1.89 |
| 74LS40 | .39 | 74LS138 | 1.25 | 74LS253 | 1.75 |
| 74LS42 | 1.25 | 74LS139 | 1.25 | 74LS257 | 1.75 |
| 74LS47 | 1.25 | 74LS151 | 1.25 | 74LS260 | .55 |
| 74LS51 | .29 | | | 74LS279 | .79 |
| 74LS55 | .29 | | | 74LS367 | .99 |
| 74LS73 | .49 | | | 74LS368 | .99 |
| 74LS74 | .49 | | | 74LS670 | 3.95 |

# BUGBOOK®

Continuing Education Series



**BUGBOOKS I and II** — $17.00 per set
by Peter R. Rony, David G. Larsen, WB4HYJ
Sold as a set these two books outline over 90 experiments designed to teach the reader all he will need to know about TTL logic chips to use them in conjunction with microprocessor systems. You'll learn about the basic concepts of digital electronics including gates, flip-flops, latches, buses, decoders, multiplexers, demultiplexers, LED displays, RAM's, ROM's, and much, much more.

**BUGBOOK IIa** — $5.00
by Peter R. Rony, David G. Larsen, WB4HYJ
This volume will introduce you to the fabulous UART chip — that all important interface between data terminals, etc., and your microcomputer. It also covers current loops, and the RS 232C interface standard. Particularly recommended for any RTTY enthusiast.

**THE 555 TIMER APPLICATIONS SOURCEBOOK WITH EXPERIMENTS** — $6.95
by Howard M. Berlin
This book shows you what the 555 timer is and how to use it. Included are over 100 various design techniques, equations and graphs to create "ready-to-go" timers, generators, power supplies, measurement and control circuits, party games, circuits for the home and automobile, photography, music and Amateur Radio.

**BUGBOOK III** — $15.00
by Peter R. Rony, David G. Larsen, WB4HYJ, Jonathan A. Titus
Here is the book that puts it all together. Besides having much valuable text there are a series of experiments in which the reader completely explores the 8080 chip pin by pin and introduces you to the Mark 80 microcomputer, a unique easily interfaced system. It is recommended that you have the background on the BUGBOOKS I & II before proceeding with BUGBOOK III.

**INSTRUCTOR'S MANUAL** — $3.00
Necessary for instruction of Bugbook I and II. Answers questions regarding experiments, suggestions for further reading, philosophy of authors approach to digital electronics. A must for self-teaching individuals.

**BUGBOOK V and VI** — $19.00 per set
by David G. Larsen, Peter R. Rony, Jonathan A. Titus
Experiments in digital electronics, 8080A microcomputer programming and 8080A microcomputer interfacing. An integrated approach to self-instructed basic digital electronics, breadboarding and 8080A interfacing/programming. Bugbook VI integrates the digital concepts of Bugbook V into a treatment of 8080A microcomputer programming and interfacing. Detail & laboratory experiments included with each book.

**OP AMP MANUAL** — $9.00
An experiment guide to application of operational amplifiers. Over 25 experiments on all phases of Op Amps.

**DBUG** — $5.00
8080 interpretive debugger. A program for entering, debugging and storing assembly language programs.

**CMOS-M — DESIGNERS PRIMER AND HANDBOOK** — $6.00
Starts at basic structure of CMOS devices through integration into MSI.

**COMPLETE MANUAL FOR DIGITAL CLOCKS** by John Weiss and John Brooks
Familiarizes technician or hobbyist with basic theories behind digital clocks. Includes trouble shooting guides, basic characteristics of clocks, soldering techniques, clock component data sheets and construction tips. $3.95

# DISCRETE LEDS

| 125" dia. | | | .190" dia. | | |
|---|---|---|---|---|---|
| XC209 | Red | 5/$1 | XC111 | Red | 10/$1 |
| XC209 | Green | 4/$1 | XC111 | Green | 4/$1 |
| XC209 | Orange | 4/$1 | XC111 | Yellow | 4/$1 |
| XC209 | Yellow | 4/$1 | XC111 | Orange | 4/$1 |

| .200" dia | | | .185" dia. | | | .200" dia. | | |
|---|---|---|---|---|---|---|---|---|
| XC22 | Red | 5/$1 | XC526 | Red | 5/$1 | XC556 | Red | 5/$1 |
| XC22 | Green | 4/$1 | XC526 | Red | 100/$8 | XC556 | Red | 100/$8 |
| XC22 | Yellow | 4/$1 | XC526 | Green | 4/$1 | XC556 | Green | 4/$1 |
| XC22 | Orange | 4/$1 | XC526 | Yellow | 4/$1 | XC556 | Yellow | 4/$1 |
| SSL-22 | RT | 4/$1 | XC526 | Clear | 4/$1 | XC556 | Orange | 4/$1 |
| | | | | | | XC556 | Clear | 7/$1 |

.085" dia. MV50 - Red - 6/$1

INFRA-RED LED ¼" x ¼" x 1/16" Flat 5/$1.00

# SALE — DISPLAY LEDS — SALE

| TYPE | POLARITY | HT | PRICE | TYPE | POLARITY | HT | PRICE |
|---|---|---|---|---|---|---|---|
| MAN 1 | Common Anode-red | .270 | 2.95 | MAN 6680 | Common Cathode-orange | .560 | .79 |
| MAN 2 | 5 x 7 Dot Matrix-red | .300 | 4.95 | MAN 6710 | Common Anode-red-D.D. | .560 | .79 |
| MAN 3 | Common Cathode-red | .125 | 4/$1 | MAN 6730 | Common Anode-red ±1 | .560 | .79 |
| MAN 4 | Common Cathode-red | .187 | 1.95 | MAN 6740 | Common Cathode-red-D.D. | .560 | .79 |
| MAN 52 | Common Anode-green | .300 | 1.00 | MAN 6750 | Common Cathode-red ±1 | .560 | .79 |
| MAN 71 | Common Anode-red | .300 | 1.25 | MAN 6760 | Common Anode-red | .560 | .79 |
| MAN 72 | Common Anode-red | .300 | .79 | MAN 6780 | Common Cathode-red | .560 | .79 |
| MAN 74 | Common Cathode-red | .300 | 1.50 | DL701 | Common Anode-red ±1 | .300 | 1.00 |
| MAN 81 | Common Anode-yellow | .300 | .79 | DL702 | Common Cathode-red | .300 | 1.25 |
| MAN 82 | Common Anode-yellow | .300 | .79 | DL704 | Common Cathode-red | .300 | .99 |
| MAN 84 | Common Cathode-yellow | .300 | .79 | DL707 | Common Anode-red | .300 | .99 |
| MAN 3620 | Common Anode-orange | .300 | .79 | DL741 | Common Anode-red | .600 | 1.49 |
| MAN 3630 | Common Anode-orange ±1 | .300 | 1.35 | DL746 | Common Anode-red ±1 | .630 | 1.95 |
| MAN 3640 | Common Cathode-orange | .300 | .79 | DL747 | Common Anode-red | .600 | 1.49 |
| MAN 4610 | Common Anode-orange | .300 | .79 | DL749 | Common Cathode-red ±1 | .630 | 1.95 |
| MAN 4640 | Common Cathode-orange | .400 | .79 | DL750 | Common Cathode-red | .600 | 1.49 |
| MAN 4710 | Common Anode-red ±1 | .400 | .79 | DL33B | Common Cathode-red | .110 | 3/1.00 |
| MAN 4730 | Common Anode-red | .400 | 1.00 | FND70 | Common Cathode (FND359) | .250 | .69 |
| MAN 4740 | Common Cathode-red | .400 | .79 | FND503 | Common Cathode (FND500) | .500 | .99 |
| MAN 4810 | Common Anode-yellow | .400 | 1.00 | FND507 | Common Anode (FND510) | .500 | .99 |
| MAN 6610 | Common Anode-orange-D.D. | .560 | .79 | 5082-7300 | 4 x 7 Sgl. Digit-RHDP | .600 | 19.95 |
| MAN 6630 | Common Anode-orange | .560 | .79 | 5082-7302 | 4 x 7 Sgl. Digit-LHDP | .600 | 19.95 |
| MAN 6640 | Common Cathode-orange-D.D. | .560 | .79 | 5082-7304 | Overrange character (±1) | .600 | 15.00 |
| MAN 6650 | Common Cathode-orange ±1 | .560 | .79 | 5082-7340 | 4 x 7 Sgl. Digit-Hexadecimal | .600 | 22.50 |
| MAN 6660 | Common Anode-orange | .560 | .79 | | | | |

# RCA LINEAR

| | | | |
|---|---|---|---|
| CA3013 | 2.15 | CA3082 | 2.00 |
| CA3023 | 2.56 | CA3083 | 1.60 |
| CA3035 | 2.48 | CA3086 | .85 |
| CA3039 | 1.35 | CA3089 | 3.75 |
| CA3046 | 1.30 | CA3091 | 3.50 |
| CA3053 | 1.50 | CA3102 | 2.95 |
| CA3059 | 3.25 | CA3123 | 2.15 |
| CA3060 | 3.25 | CA3130 | 1.39 |
| CA3080 | .85 | CA3140 | 1.25 |
| CA3081 | 2.00 | CA3401 | .49 |
| | | CA3600 | 1.75 |

# EXAR

XR-2206KB Kit $19.95 — XR-2206KA Kit $14.95

| WAVEFORM GENERATORS | | MISCELLANEOUS | | TIMERS | |
|---|---|---|---|---|---|
| XR-205 | $8.40 | XR-2211CP | $6.70 | XR-555CP | $ .49 |
| XR-2206CP | 5.50 | XR-4136 | 2.00 | XR-320P | 1.55 |
| XR-2207CP | 3.85 | XR-1468 | 3.85 | XR-556CP | 1.85 |
| **STEREO DECODERS** | | XR-1488 | 5.80 | XR-2556CP | 3.20 |
| XR-1310CP | $3.20 | XR-1489 | 4.80 | XR-2240CP | 4.80 |
| XR-1310EP | 3.20 | XR-2208 | 5.20 | **PHASE LOCKED LOOPS** | |
| XR-1800P | 3.20 | | | XR-210 | 5.20 |
| XR-2567 | 2.99 | | | XR-215 | 6.60 |
| | | | | XR-567CP | 1.95 |
| | | | | XR-567CT | 1.70 |

# IC SOLDERTAIL — LOW PROFILE (TIN) SOCKETS

| | 1-24 | 25-49 | 50-100 | | 1-24 | 25-49 | 50-100 |
|---|---|---|---|---|---|---|---|
| 8 pin | $.17 | .16 | .15 | 24 pin | $.38 | .37 | .36 |
| 14 pin | .20 | .19 | .18 | 28 pin | .45 | .44 | .43 |
| 16 pin | .22 | .21 | .20 | 36 pin | .60 | .59 | .58 |
| 18 pin | .29 | .28 | .27 | 40 pin | .63 | .62 | .61 |
| 22 pin | .37 | .36 | .35 | | | | |

**SOLDERTAIL STANDARD (TIN)**

| | 1-24 | 25-49 | 50-100 | | 1-24 | 25-49 | 50-100 |
|---|---|---|---|---|---|---|---|
| 14 pin | $.27 | .25 | .24 | 28 pin | $ .99 | .90 | .81 |
| 16 pin | .30 | .27 | .25 | 36 pin | 1.39 | 1.26 | 1.15 |
| 18 pin | .35 | .32 | .30 | 40 pin | 1.59 | 1.45 | 1.30 |
| 24 pin | .49 | .45 | .42 | | | | |

**SOLDERTAIL STANDARD (GOLD)**

| | 1-24 | 25-49 | 50-100 | | 1-24 | 25-49 | 50-100 |
|---|---|---|---|---|---|---|---|
| 8 pin | $.30 | .27 | .24 | 24 pin | $. 70 | .63 | .57 |
| 14 pin | .35 | .32 | .29 | 28 pin | 1.10 | 1.00 | .90 |
| 16 pin | .38 | .35 | .32 | 36 pin | 1.75 | 1.40 | 1.26 |
| 18 pin | .52 | .47 | .43 | 40 pin | 1.75 | 1.59 | 1.45 |

**WIRE WRAP SOCKETS (GOLD) LEVEL #3**

| | 1-24 | 25-49 | 50-100 | | 1-24 | 25-49 | 50-100 |
|---|---|---|---|---|---|---|---|
| 8 pin | $.40 | .38 | .35 | 22 pin | .95 | .85 | .75 |
| 10 pin | .45 | .41 | .37 | 24 pin | $1.05 | .95 | .85 |
| 14 pin | .39 | .38 | .37 | 28 pin | 1.40 | 1.25 | 1.10 |
| 16 pin | .43 | .42 | .41 | 36 pin | 1.59 | 1.45 | 1.30 |
| 18 pin | .75 | .68 | .62 | 40 pin | 1.75 | 1.55 | 1.40 |

# 50 PCS. RESISTOR ASSORTMENTS $1.75 PER ASST.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ASST. 1 | 5 ea. | 10 OHM / 27 OHM | 12 OHM / 33 OHM | 15 OHM / 39 OHM | 18 OHM / 47 OHM | 22 OHM / 56 OHM | 1/4 WATT 5% — 50 PCS. |
| ASST. 2 | 5 ea. | 68 OHM / 180 OHM | 82 OHM / 220 OHM | 100 OHM / 270 OHM | 120 OHM / 330 OHM | 150 OHM / 390 OHM | 1/4 WATT 5% = 50 PCS. |
| ASST. 3 | 5 ea. | 470 OHM / 1.2K | 560 OHM / 1.5K | 680 OHM / 1.8K | 820 OHM / 2.2K | 1K / 2.7K | 1/4 WATT 5% = 50 PCS. |
| ASST. 4 | 5 ea. | 3.3K / 8.2K | 3.9K / 10K | 4.7K / 12K | 5.6K / 15K | 6.8K / 18K | 1/4 WATT 5% = 50 PCS. |
| ASST. 5 | 5 ea. | 22K / 56K | 27K / 68K | 33K / 82K | 39K / 100K | 47K / 120K | 1/4 WATT 5% = 50 PCS. |
| ASST. 6 | 5 ea. | 150K / 390K | 180K / 470K | 220K / 560K | 270K / 680K | 330K / 820K | 1/4 WATT 5% = 50 PCS. |
| ASST. 7 | 5 ea. | 1M / 2.7M | 1.2M / 3.3M | 1.5M / 3.9M | 1.8M / 4.7M | 2.2M / 5.6M | 1/4 WATT 5% = 50 PCS. |

**ASST. 8R** Includes Resistor Assortments 1-7 (350 PCS.) **$9.95 ea.**

# WIRE-WRAP KIT — WK-2-W

**WRAP • STRIP • UNWRAP**

- Tool for 30 AWG Wire
- Roll of 50 Ft. White or Blue 30 AWG Wire
- 50 pcs. each 1", 2", 3" & 4" lengths — pre-stripped wire.

**$11.95**

## WIRE WRAP TOOL WSU-30

WRAP • STRIP • UNWRAP - $5.95

## WIRE WRAP WIRE — 30 AWG

25 ft. min. $1.25 50 ft. $1.95 100 ft. $2.95 1000 ft. $15.00
SPECIFY COLOR — White - Yellow - Red - Green - Blue - Black

## WIRE DISPENSER — WD-30

- 50 ft. roll 30 AWG KYNAR wire wrap wire **$3.45 ea.**
- Cuts wire to desired length
- Strips 1" of insulation **Specify** — Blue-Yellow-White-Red

**REPLACEMENT DISPENSER SPOOLS FOR WD 30**
Specify blue, yellow, white or red **$1.98/spool**

# Prime Stock Inventory Clearance — STEWART WARNER — Special Offer

| P/N | 10(min.) | 100 | 1000 | P/N | 10(min.) | 100 | 1000 |
|---|---|---|---|---|---|---|---|
| SW7401 | $1.20/lot | $9.00/lot | $80.00/lot | SW7482 | $4.90/lot | 44.00/lot | 430.00/lot |
| SW7407 | 2.20 | 19.00 | 180.00 | SW7483 | 3.50 | 31.00 | 300.00 |
| SW7416 | 2.20 | 19.00 | 180.00 | SW7486 | 2.20 | 19.00 | 180.00 |
| SW7417 | 2.20 | 19.00 | 180.00 | SW7491 | 3.50 | 31.00 | 300.00 |
| SW7420 | 1.30 | 11.00 | 100.00 | SW7494 | 3.50 | 31.00 | 300.00 |
| SW7423 | 2.20 | 19.00 | 180.00 | SW7495 | 3.50 | 31.00 | 300.00 |
| SW7425 | 1.70 | 14.00 | 130.00 | SW7496 | 3.50 | 31.00 | 300.00 |
| SW7427 | 1.50 | 12.00 | 110.00 | SW74100 | 6.90 | 66.00 | 650.00 |
| SW7430 | 1.50 | 12.00 | 110.00 | SW74104 | 2.20 | 19.00 | 180.00 |
| SW7432 | 1.50 | 12.00 | 110.00 | SW74105 | 2.20 | 19.00 | 180.00 |
| SW7437 | 1.50 | 12.00 | 110.00 | SW74107 | 2.20 | 19.00 | 180.00 |
| SW7438 | 1.50 | 12.00 | 110.00 | SW74121 | 2.20 | 19.00 | 180.00 |
| SW7440 | 1.20 | 9.00 | 80.00 | SW74123 | 2.50 | 22.00 | 200.00 |
| SW7443 | 3.50 | 31.00 | 300.00 | SW74145 | 5.50 | 51.00 | 500.00 |
| SW7444 | 3.50 | 31.00 | 300.00 | SW74150 | 6.90 | 66.00 | 650.00 |
| SW7445 | 3.50 | 31.00 | 300.00 | SW74151 | 3.50 | 31.00 | 300.00 |
| SW7446 | 4.00 | 36.00 | 350.00 | SW74153 | 4.00 | 36.00 | 350.00 |
| SW7450 | 1.20 | 9.00 | 80.00 | SW74156 | 4.00 | 36.00 | 350.00 |
| SW7453 | 1.20 | 9.00 | 80.00 | SW74180 | 4.50 | 41.00 | 400.00 |
| SW7454 | 1.20 | 9.00 | 80.00 | SW74181 | 9.90 | 96.00 | 950.00 |
| SW7460 | 1.20 | 9.00 | 80.00 | SW74182 | 4.50 | 41.00 | 400.00 |
| SW7472 | 2.20 | 19.00 | 180.00 | SW9601 | 2.50 | 22.00 | 200.00 |
| SW7475 | 3.50 | 31.00 | 300.00 | SW9602 | 4.90 | 44.00 | 430.00 |
| SW7480 | 2.90 | 26.00 | 250.00 | | | | |

**Pre-tubed • No mixing or combining prices**

# TV GAME CHIP SET — $7.95

Includes AY-3-8500-1 Chip and 2.010 mhz crystal
(2.010 crystal — $.99 ea/AY-3-8500-1 Chip — $7.50 ea.)

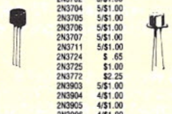
# ZENERS — DIODES — RECTIFIERS

| TYPE | VOLTS | W | PRICE | TYPE | VOLTS | W | PRICE |
|---|---|---|---|---|---|---|---|
| 1N746 | 3.3 | 400m | 4/1.00 | 1N4005 | 600 PIV | 1 AMP | 10/1.00 |
| 1N751A | 5.1 | 400m | 4/1.00 | 1N4006 | 800 PIV | 1 AMP | 10/1.00 |
| 1N752 | 5.6 | 400m | 4/1.00 | 1N4007 | 1000 PIV | 1 AMP | 10/1.00 |
| 1N753 | 6.2 | 400m | 4/1.00 | 1N3600 | 50 | 200m | 6/1.00 |
| 1N754 | 6.8 | 400m | 4/1.00 | 1N4148 | 75 | 10m | 15/1.00 |
| 1N959 | 8.2 | 400m | 8/1.00 | 1N4154 | 35 | 10m | 12/1.00 |
| 1N965B | 15 | 400m | 4/1.00 | 1N4305 | 75 | 25m | 20/1.00 |
| 1N5232 | 5.6 | 500m | 28 | 1N4734 | 5.6 | 1w | 28 |
| 1N5234 | 6.2 | 500m | 28 | 1N4735 | 6.2 | 1w | 28 |
| 1N5235 | 6.8 | 500m | 28 | 1N4736 | 6.8 | 1w | 28 |
| 1N5236 | 7.5 | 500m | 28 | 1N4738 | 8.2 | 1w | 28 |
| 1N456 | 25 | 40m | 6/1.00 | 1N4742 | 12 | 1w | 28 |
| 1N458 | 150 | 7m | 6/1.00 | 1N4744 | 15 | 1w | 28 |
| 1N485A | 180 | 10m | 6/1.00 | 1N1183 | 50 PIV | 35 AMP | 1.60 |
| 1N4001 | 50 PIV | 1 AMP | 12/1.00 | 1N1184 | 100 PIV | 35 AMP | 1.70 |
| 1N4002 | 100 PIV | 1 AMP | 12/1.00 | 1N1185 | 150 PIV | 35 AMP | 1.50 |
| 1N4003 | 200 PIV | 1 AMP | 12/1.00 | 1N1186 | 200 PIV | 35 AMP | 1.80 |
| 1N4004 | 400 PIV | 1 AMP | 12/1.00 | 1N1188 | 400 PIV | 35 AMP | 3.00 |

# SCR AND FW BRIDGE RECTIFIERS

| | | | |
|---|---|---|---|
| C36D | 15A @ 400V | SCR | $1.95 |
| C38M | 35A @ 200V | SCR | 1.95 |
| 2N2328 | 1.6A @ 200V | SCR | .50 |
| MDA 980-1 | 12A @ 50V | FW BRIDGE REC. | 1.95 |
| MDA 980-3 | 12A @ 200V | FW BRIDGE REC. | 1.95 |

# TRANSISTORS



| | | | | | |
|---|---|---|---|---|---|
| MPS A05 | .30 | | | PN4250 | 4/$1.00 |
| MPS A06 | 5/$1.00 | | | 2N4400 | 4/$1.00 |
| 2N918 | 4/$1.00 | | | 2N4401 | 4/$1.00 |
| 2N2219A | 3/$1.00 | 2N3638 | 5/$1.00 | 2N4402 | 4/$1.00 |
| 2N2221 | 4/$1.00 | 2N3702 | 5/$1.00 | 2N4403 | 4/$1.00 |
| 2N2222A | 5/$1.00 | 2N3704 | 5/$1.00 | 2N4409 | 5/$1.00 |
| 2N2369 | 5/$1.00 | 2N3705 | 5/$1.00 | 2N5086 | 4/$1.00 |
| 2N2369A | 4/$1.00 | 2N3706 | 5/$1.00 | 2N5087 | 4/$1.00 |
| 2N2484 | 4/$1.00 | 2N3707 | 5/$1.00 | 2N5088 | 4/$1.00 |
| 2N2906A | 4/$1.00 | 2N3711 | 5/$1.00 | 2N5089 | 4/$1.00 |
| 2N2907A | 5/$1.00 | 2N3724 | $ .65 | 2N5129 | 5/$1.00 |
| 2N2925 | 5/$1.00 | 2N3725 | $1.00 | 2N5138 | 5/$1.00 |
| 2N3035 | 2/$1.00 | 2N3772 | $2.25 | 2N5139 | 5/$1.00 |
| 2N3055 | $ .89 | 2N3903 | 5/$1.00 | 2N5209 | 5/$1.00 |
| MJE2955 | $1.25 | 2N3904 | 4/$1.00 | 2N5210 | 5/$1.00 |
| MJE3055 | $1.00 | 2N3905 | 4/$1.00 | 2N5432 | $2.00 |
| 2N3392 | 5/$1.00 | 2N3906 | 4/$1.00 | 2N5449 | 3/$1.00 |
| 2N3398 | 5/$1.00 | 2N4013 | 3/$1.00 | 2N5951 | 5/$1.00 |
| PN3567 | 3/$1.00 | 2N4014 | 3/$1.00 | C106B1SCR | 2/$1.00 |
| PN3568 | 4/$1.00 | 2N4123 | 6/$1.00 | 40409 | $1.75 |
| PN3569 | 4/$1.00 | PN4249 | 4/$1.00 | 40410 | $1.75 |

# CAPACITOR CORNER

**50 VOLT CERAMIC DISC CAPACITORS**

| | 1-9 | 10-49 | 50-100 | | 1-9 | 10-49 | 50-100 |
|---|---|---|---|---|---|---|---|
| 10 pf | .05 | .04 | .03 | .001µF | .05 | .04 | .035 |
| 22 pf | .05 | .04 | .03 | .0047µF | .05 | .04 | .035 |
| 47 pf | .05 | .04 | .03 | 01µF | .05 | .04 | .035 |
| 100 pf | .05 | .04 | .03 | .022µF | .06 | .05 | .04 |
| 220 pf | .05 | .04 | .03 | .047µF | .06 | .05 | .04 |
| 470 pf | .05 | .04 | .035 | .1µF | .12 | .09 | .075 |
| **100 VOLT MYLAR FILM CAPACITORS** | | | | | | | |
| .001mf | .12 | .10 | .07 | .022mf | .13 | .11 | .08 |
| .0022 | .12 | .10 | .07 | .047mf | .21 | .17 | .13 |
| .0047mf | .12 | .10 | .07 | .1mf | .27 | .23 | .17 |
| .01mf | .12 | .10 | .07 | .22mf | .33 | .27 | .22 |
| **+20% DIPPED TANTALUMS (SOLID) CAPACITORS** | | | | | | | |
| .1/35V | .28 | .23 | .17 | 1.5/35V | .30 | .26 | .21 |
| .15/35V | .28 | .23 | .17 | 2.2/25V | .31 | .27 | .22 |
| .22/35V | .28 | .23 | .17 | 3.3/25V | .31 | .27 | .22 |
| .33/35V | .28 | .23 | .17 | 4.7/25V | .32 | .28 | .23 |
| .47/35V | .28 | .23 | .17 | 6.8/25V | .36 | .31 | .25 |
| .68/35V | .28 | .23 | .17 | 10/25V | .40 | .35 | .29 |
| 1.0/35V | .28 | .23 | .17 | 15/25V | .63 | .50 | .40 |
| **MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS** | | | | | | | |
| Axial Lead | | | | Radial Lead | | | |
| .47/50V | .15 | .13 | .10 | .47/25V | .15 | .13 | .10 |
| 1.0/50V | .16 | .14 | .11 | .47/50V | .16 | .14 | .11 |
| 3.3/50V | .14 | .13 | .10 | 1.0/16V | .15 | .13 | .10 |
| 4.7/25V | .16 | .14 | .12 | 1.0/25V | .16 | .14 | .11 |
| 10/25V | .15 | .13 | .10 | 1.0/50V | .16 | .14 | .11 |
| 10/50V | .16 | .14 | .12 | 4.7/16V | .15 | .13 | .10 |
| 22/25V | .17 | .15 | .12 | 4.7/25V | .15 | .13 | .10 |
| 22/50V | .24 | .20 | .18 | 4.7/50V | .16 | .14 | .11 |
| 47/25V | .19 | .17 | .15 | 10/16V | .14 | .12 | .09 |
| 47/50V | .25 | .21 | .19 | 10/25V | .15 | .13 | .10 |
| 100/25V | .24 | .20 | .18 | 10/50V | .16 | .14 | .12 |
| 100/50V | .35 | .30 | .28 | 47/50V | .24 | .21 | .19 |
| 220/25V | .32 | .28 | .25 | 100/16V | .19 | .15 | .14 |
| 220/50V | .45 | .41 | .38 | 100/25V | .24 | .20 | .18 |
| 470/25V | .33 | .29 | .27 | 100/50V | .35 | .30 | .28 |
| 1000/16V | .55 | .50 | .45 | 220/16V | .23 | .17 | .16 |
| 2200/16V | .70 | .62 | .55 | 470/25V | .31 | .28 | .26 |

**$5.00 Minimum Order — U.S. Funds Only**
**California Residents — Add 6% Sales Tax**
**Spec Sheets - 25¢ — Send 35¢ Stamp for 1978 Catalog**
**Dealer Information Available**

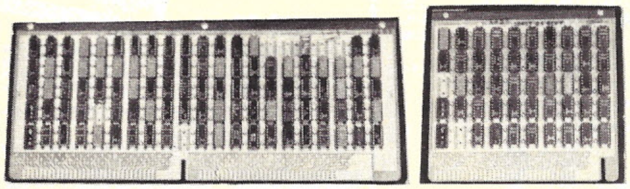
NOW OPEN SATURDAYS

**James ELECTRONICS**

1021-A HOWARD AVE., SAN CARLOS, CA. 94070
PHONE ORDERS WELCOME — (415) 592-8097
Advertised Prices Good Thru April

**1978 CATALOG NOW AVAILABLE**

# WIRE WRAP BOARDS LOADED WITH 7400 SERIES ICs



Since last summer, we have been selling 2 wire wrap boards, Our Stock No. 6558K with approximately 100 sockets, and our Stock No. 6559K with approximately 45 sockets. These have been successful, based on your orders and reorders. We now have the same boards, but with the sockets still containing the original SN7400 series ICs that were used in the computer that these boards were designed for. We checked the value of these ICs, against the lowest price ICs in several Electronics magazine, and found that at the lowest possible surplus prices, the values of the ICs on the 100 socket board ran to over $40.00. A sample of some of the chips on the board we looked at are as f ollows: 74H87, 7486, 74107, 7451, 7400, 7404, 7495, 7493, 7492, 74193, 7489 and many others, to numerous to mention. Also on some boards, are a few linears, and phase locked loops. Not everyone needs every chip, but if you are working at all with TTL, this is a great opportunity to get an inventory of the most useful chips at a ridiculous price. We are selling the 100 socket board with about 100 chips, for $10.00 more than the board itself, and the 45 socket chip for $5.00 more than the board itself. We will also include with each board, 2 edge connectors with the 100 socket board, and 1 edge connector with the 45 socket board.

| | | | |
|---|---|---|---|
| STOCK NO.6558K | 75 to 100 socket board | $18.75 ea. | 2/35.00 |
| STOCK NO.6559K | 45 to 50 socket board | $11.75 | 2/22.00 |
| STOCK NO.6749K | 75 to 100 socket wire wrap board with ICs and edge connectors | $28.75 ea. | 2/55.00 |
| STOCK NO.6750K | 45 to 50 socket wire wrap board with ICs and edge connector | $16.75 ea. | 2/32.00 |
| STOCK NO.6603K | Edge connector for either board | $2.00 ea | 3/5.00 |

# VIDEO MONITORS

For the past several months, we have been selling VIDEO MONITORS through these pages. We have now exhausted supply of complete working monitors, and have left a few lots that may have some interest to our readers. We have a limited number of MONITORS that are in working condition, but lack a picture tube. We will sell these monitors for about half of what we sold the complete monitor for. The second lot does not have picture tubes, but has all other parts, but no guaranteee that all parts are in working condition. Note in some cases in each of the above lots, a picture tube may be included, but it will have some burnt spots on the screen. No guarantee on which ones these will be.

STOCK NO. 5585K Working monitors without picture tubes $49.95 ea.

STOCK NO. 5586K Monitors without picture tubes, need other repairs $29.95 ea.

# NEW GENERL PURPOSE TRANSFORMERS

Primary 115 VAC. Sec 1. 12.6 VAC ct. @ 1.0 A. Sec. 2. 33 VAC tapped @ 13.1 @ 1.0 A. Sec. 3 140 VAC @ 100 ma.

STOCK NO.6751K 4 1/16"x2 1/16"x2 5/8" Wt. 2 lbs. $2.95 ea. 2/5.00

---

Primary 115 VAC. Sec. 1. 23VAC @ 9 A. Sec. 2 14 VAC @ 20 A. Sec. 3 11.5 VAC @ 4.8 A. Sec.4, 125 VAC @ 1.5 A. Also has a second primary winding, so that all secondaries may be obtained with a 220 input, or if 2 primaries are placed in series on 115 VAC, all secondary voltages are divided by 2.

STOCK NO.6675N 5¾x5"x4½" 16 lbs. $17.95 ea. 2/34.00

---

Primary 115 with tap to give seconday of either 8.6 or 9.5 VAC @ 2 A. Ideal for TTL power supply.

STOCK NO.6726N 3 1/8"x2 7/8" 2½" Wt. 3½ lbs. $3.95 ea. 2/7.00

---

Primary 115 VAC. Sec. 1. 30 VAC. @ 2 A. Sec. 2. 16.5 VAC @ 1.5 A. Sec.3, 16 VAC @ 3.5 A. Sec. 4. 9.5 VAC @3.5 A.

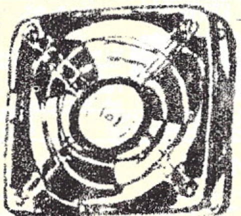
STOCK NO.6677N 4½"x4¼x4" 10 lbs. $10.95 ea. 2/20.00

# VIDEOCUBE

THE COMPUTER / TV INTERFACE

VIDEOCUBE is a self contained oscillator and modulator, which allows easy interface with any device having a video output, and a standard TV set. When properly used, the output of your video camera, video game or video output of your computer is displayed on channel 3. Easy switching from TV to monitor. The VIDEOCUBE was completely described in August issue of RADIO-ELECTRONICS. We supply a reprint of this article. Has FCC approval for radiation.

STOCK NO. 5500K Complete kit of parts with data. $13.95 2/26.00



# ROTRON WHISPER FANS

KEEP YOUR EQUIPMENT COOL, with ROTRON WHISPER FANS. 115 VAC. 7 Watts. These fans have been removed from equipment, and are fully guaranteed. 4½"x4½"x1½"

STOCK NO.5520K $6.95 ea. 2/12.00

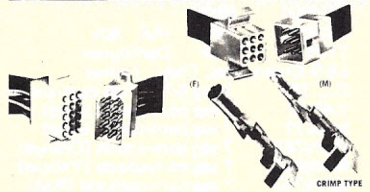
MAIL ORDER

**DELTA ELECTRONICS**
**P.O. BOX 2, 7 Oakland St.**
**Amesbury, Mass. 01913**

Over the counter salesrooms

**ATLANTA GA**
DELTA ELECTRONIC HOBBIES
5151 Buford Hwy.
Doraville, Atlanta, Ga.

**BOSTON MASS.**
DELTA ELECTRONICS
WAREHOUSE OUTLET
590 Commonwealth Ave.
Boston, Mass. D13

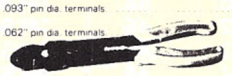



## NYLON CONNECTORS

Mfg by Molex

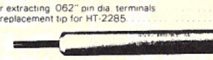
| Complete Connectors Per Pkg. | Type No. | Class | Description | Price Ea. Pkg. |
|---|---|---|---|---|
| 5 | 1625-1PRT | Min.(.062') | 1 Circuit | $1.75 |
| 3 | 1625-2PRT | " | 2 Circuit | 1.90 |
| 3 | 1625-3PRT | " | 3 Circuit | 2.10 |
| 2 | 1625-4PRT | " | 4 Circuit | 2.10 |
| 2 | 1625-5PRT | " | 5 Circuit | 2.20 |
| 2 | 1625-6PRT | " | 6 Circuit | 2.35 |
| 1 | 1649-8PRT | " | 8 Circuit | 1.55 |
| 1 | 1625-9PRT | " | 9 Circuit | 1.75 |
| 1 | 1625-12PRT | " | 12 Circuit | 1.90 |
| 1 | 1625-15PRT | " | 15 Circuit | 2.30 |
| 1 | 1625-24PRT | " | 24 Circuit | 3.25 |
| 1 | 1772-36PRT | " | 36 Circuit | 4.55 |
| 5 | 1619PRT | Std.(.093") | 1 Circuit | 1.75 |
| 3 | 1545PRT | " | 2 Circuit | 1.90 |
| 3 | 1396PRT | " | 3 Circuit | 2.10 |
| 2 | 1490PRT | " | 4 Circuit | 2.10 |
| 2 | 1653PRT | " | 5 Circuit | 2.20 |
| 2 | 1261PRT | " | 6 Circuit | 2.35 |
| 1 | 1292PRT | " | 9 Circuit | 1.80 |
| 1 | 1360PRT | " | 12 Circuit | 1.90 |
| 1 | 1375PRT | " | 15 Circuit | 2.45 |

**Prototype hand tools** combine efficiency with economy. Ideal for prototype or limited production runs.
HT-1919 for .093" pin dia. terminals ........ $8.95 each
HT-1921 for .062" pin dia. terminals ........ $8.95 each



**Econo-Extractor** removes terminal from nylon connector housing with smoothness and ease.
HT-2054 for extracting .093" pin dia. terminals ........ $2.25 each
HT-2023 for extracting .062" pin dia. terminals ........ $2.25 each

**Deluxe ejector tools,** spring-loaded for simple, efficient removal of terminal from nylon connector housing-extracts either male or female terminals of same pin diameter.
HT-2038 for extracting .093" pin dia. terminals ........ $6.70 each
HT-1010-2B2 Replacement tip for HT-2038 ........ $2.50 each
HT-2285 for extracting .062" pin dia. terminals ........ $6.70 each
HT-1672-3 replacement tip for HT-2285 ........ $2.50 each



### INTEGRATED CIRCUITS

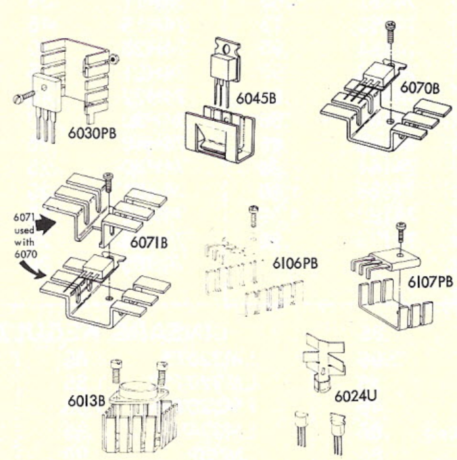
| | |
|---|---|
| 555 Timer 8 pin mini-DIP | .49 |
| 741 Compensated OP-Amp 8 pin DIP | .37 |
| LM 1889N RF Video Modulator | 7.45 |
| CA3130 Bipolar/Mos-FET Op Amp | 1.19 |
| CA3140 MOS-FET Op Amp, Bi-polar out | .99 |
| LM3909 Lo Voltage Led Pulser | 1.50 |
| LM3911 Temp Control CHIP | 1.50 |
| Signetics 2504TA 1024 bit S.R. memory (1404A) | $1.75 |
| MCM 6571P Character Generator | 11.95 |
| MCM6571AP Character Generator | 11.95 |
| MCM6575P Character generator | $14.88 |
| 50240 Top octave generator | $14.95 |
| LM399H Temp Stabilized Zener | 5.95 |
| AF100-1CJ Active Filter, State Variable | 7.50 |
| LM2907N Tachometer F/V Converter | 2.65 |
| LM1812N Ultra-sonic Transceiver | 9.15 |
| LM1815 Adaptive Sense Amp for Tachometer | 5.73 |
| TL170 TO-92 Hall effect switch w/spec sheets | 1.25 |
| MC14409P Telephone Rotary Pulser | 10.98 |
| MC14419P Touch Pad Converter for 14409 | 4.25 |
| MC14411P Baud Rate Generator | 11.98 |
| MC14412VP CMOS Modem Chip | 16.95 |
| MM57109N Number Cruncher Micro | 18.95 |
| 74C915 7 Segment to BCD Converter | 2.99 |
| 74C922 16 Key Keyboard Encoder | 6.35 |
| 74C923 20 key Keyboard Encoder | 6.45 |
| 74C925 4 Decade Counter w/latches | 12.00 |
| 74C926 4 Decade Counter w/carry | 12.00 |
| 74C935-1 3½ Digit DVM CMOS Chip | 16.98 |
| 9601 Retriggerable One shot | .50 |
| MC4015P Hi Speed quad "D" low power TTL | $1.00 |

### ADJUSTABLE NEGATIVE REGULATOR

LM337 is the compliment to the popular LM317 positive adjustable regulator. Capable of 1.5Amp from - 1.2V to - 37V.
LM337K.....(TO-3 Metal)........ $5.99
LM337T.....(TO-220 Plastic)........ $4.65
Specs and applications ........ 60¢

Neither snow, nor rain, nor heat,...etc,..etc, stops TRI-TEK from delivering quality parts. GIVE US A TRY!!!!!!!

## HEAT SINKS




Economical 1 piece heat sinks for plastic power parts in TO-220 and Motorola cases 77, 90, 199, and TO-126. All are black anodized aluminum. "B" series is anodized after forming. "PB" series is anodized prior to forming.

| | | | |
|---|---|---|---|
| THM 6030PB | Vertical | 25¢, 5/$1.00, 10/$1.90 |
| THM 6045B | Slip Over | 30¢, 4/$1.00, 10/$2.00 |
| THM 6070B | Extra Disipation Horiz. | 35¢, 3/$1.00, 10/$3.00 |
| THM 6071B | Top Hat for 6070 | 35¢, 3/$1.00, 10/$3.00 |
| THM 6106PB | Flat With Fingers | 30¢, 4/$1.00, 10/$2.00 |
| THM 6107PB | Smaller Size Flat | 25¢, 5/$1.00, 10/$1.90 |
| THM 6013B | TO-3 Diamond | 69¢, 4/$2.50, 10/$5.00 |
| THM 6024-U | Unfinished TO-92 | 10/$1.00, 100/$5.00 |

OPTO COUPLED TRIAC
MOC-3010 provides 115 vac full wave switching and isolation. Used alone can switch up 7.5 watts from low power inputs such as TTL logic. Drive larger triacs and control the world.
MOC-3010P........ $2.65
Specs/Apps........ .40¢

HEX DARLINGTON ARRAY
MC1413P is a 16 pin DIP package with (6) 50V 500mA Darlington pairs.
MC1413P........ $1.59
Specs/Apps........ .30¢

S-100 BUS CONNECTORS (IMSAI TYPE)
Gold, Solder tail for Mother boards $4.50, 4/$17.00
Tin-Nickel, (NASGLO) Solder tail $3.75, 4/$14.00
Gold, wire-wrap........ $4.50, 4/$17.00
Tin-Nickel, (NASGLO) wire-wrap $3.75, 4/$14.00

### I.C. SOCKETS

Lo Profile Tin Solder Tail Dip Sockets

| | | | |
|---|---|---|---|
| 8 pin | 10/$1.50 | 100/$14.00 | 1000/$120.00 |
| 14 pin | 10/$1.70 | 100/$16.00 | 1000/$140.00 |
| 16 pin | 10/$1.90 | 100/$18.00 | 1000/$160.00 |

### VOLTAGE REGULATORS

| | | |
|---|---|---|
| 7805-06-08-12-15-24 TO220 | 95¢ | 5/$4.50 |
| 7905-06-08-12-15-24 TO 220 | 95¢ | 5/$4.50 |
| 78L05A-12-15 4% 100 mA TO-92 Plastic | | 50¢ |
| 78H05KC 5V 5A TO-3 | | 9.15 |
| 78H12KC 12V 5A TO-3 | | 9.15 |
| 78H15KC 15V 5A TO-3 | | 9.15 |
| Lm317K 1.5A *Adjustable* TO-3 | | 4.99 |
| Lm317T 1.5A *Adjustable* TO-220 | | 3.99 |
| Lm317MP .5A *Adjustable* TO-202 | | 13.95 |
| TL430C Adjustable Zener-*Think About It* | | 1.50 |
| TL497C Switching Reg. & Inductor | | 9.50 |
| RCA CA 3085 100 mA *Adjustable* | | .60 |

Minature Reed relays
D.P.D.T. 12V DC 1000 Ohm coil. Only 3/8"x½"x7/8" molded case. 5/8" long flexible wire leads. Inventory reduction at large manufacturer brings you this super buy
RRY-2212G........ $1.95

SPST N.O. 5VDC 140 Ohm coil. Measures a tiny 1" long 3/8" diameter. Has flexible solid wire leads.
RRY-1105G........ 90¢

### DIODES AND BRIDGES

| | | |
|---|---|---|
| IN4003 200 V 1 amp | | 12/$1.00 |
| IN4004 400 V 1 amp | | 10/$1.00 |
| IN4148 Hi Speed Signal | 15/$1.00 | 100/$5.00 |
| D-600 115 V, 100 mA Hi Speed Signal | | 20/$1.00 |
| D2131 200 V, 25A Stud | | 85¢ |
| D2135 400 V, 25A Stud | | 1.00 |
| D2138 600 V, 25A Stud | | 1.55 |
| D3289R 200 V, 160A Stud Anode | | 5.85 |
| D3909-4 50 V, 45A Fast Recovery | | 2.00 |
| IN4732A-47A 1W 5% Zeners | | 4/$1.00 |
| 13 Assorted *Brand New* Zener Diodes | | 1.00 |
| 50V 3 amp Epoxy Bridge | | 79¢ |
| 200V 30 amp Bridge | | 2.00 |
| 600V 4 amp Epoxy Bridge | | 1.49 |
| 600V 3 amp Stud Bridge | | .89 |
| SI-2 200V, 1.5A Gold Leads | | 15/$1.00 |
| D1A-0030 30V DIAC | | 10/$1.00 |

### MISCELLANEOUS

| | |
|---|---|
| RG-174 Miniature 50 Ω coax | 50'/4.25 |
| WSU-30 Wire Wrap/unwrap tool | 5.95 |
| WSU-30M Modified Wrap/unwrap tool | 6.95 |
| BW-630 Battery Operated Wrap Tool — Free Wire with any Wrap Tool — | 34.95 |
| Miniature Square .05/100V Monolithic Cap | 10/2.00 |
| 2N4036 90V, 1A PNP Silicon TO-5 | .50 |
| 2N6101 80V, 10A NON HI GAIN TO-220 | .50 |
| SE7005 250V, ½A NPN Silicon TO-5 W/Flange | .60 |
| 6.3 VCT, 1.2A Transformer F41X | 2.49 |
| 12V, 1A Transformer with 6' Power Cord | 2.88 |

Fairchild Linear Data Book. Huge volume has 17 chapters of data, applications and definitions. Probably the biggest volume of its type today........ $4.50ppd

Fairchild Bipolar Memory Data Book. ROMS, PROMS and RAMS are covered in this work with full engineering data. Price includes shipping!........ $3.75

Raytheon Linear Data Book. Covers many of the familiar as weel as those for which Raytheon is the inovator. Contains valuable applications data as well. as complete specifications........ $2.95ppd

Raytheon Micro Computer Components Book. Covers Raytheons' offerings in the field from the micro-puter to memories and adapters........ $2.95ppd

tri-tek, inc.
7808 North 27th Avenue
Phoenix, Arizona 85021
(602) 995-9352

T1

- Please give street address for UPS shipping when possible.
- C.O.D. NO parcel post C.O.D. UPS C.O.D. Add 85¢ to order.
- Any correspondence not connected with your order, please use separate sheet and include SASE for reply.
- Orders less than $10 ($15 foreign) please add $1 handling.
- Prices are subject to change without notice.
- Any refunds will be by check, not credit vouchers.
- Terms. Check, money order, credit card. Net 30 days to rated firms, schools and government agencies.
- If we should be temporarily out of stock on an item, it will be placed on back order. If we cannot ship in 30 days, you will be notified of the expected shipping date and furnished with a postage paid card with which to cancel your order if desired.
- We pay surface shipping only in USA, Canada and Mexico.
- For premium shipping (first class, special handling, etc.) add extra. Excell will be refunded.
- Foreign orders (except Canada and Mexico) estimate and add shipping. Excess will be refunded.

**Charge card telephone orders ($20 min.) will be accepted 9-5:30 P.M. except weekends. Telephone 995-9352. No collect calls please.**

## DIODES/ZENERS

| | | | |
|---|---|---|---|
| 1N914 | 100v | 10mA | .05 |
| 1N4005 | 600v | 1A | .08 |
| 1N4007 | 1000v | 1A | .15 |
| 1N4148 | 75v | 10mA | .05 |
| 1N753A | 6.2v | z | .25 |
| 1N758A | 10v | z | .25 |
| 1N759A | 12v | z | .25 |
| 1N4733 | 5.1v | z | .25 |
| 1N5243 | 13v | z | .25 |
| 1N5244B | 14v | z | .25 |
| 1N5245B | 15v | z | .25 |

## SOCKETS/BRIDGES

| | | | | |
|---|---|---|---|---|
| 8-pin | pcb | .25 | ww | .45 |
| 14-pin | pcb | .25 | ww | .40 |
| 16-pin | pcb | .25 | ww | .40 |
| 18-pin | pcb | .25 | ww | .75 |
| 22-pin | pcb | .45 | ww | 1.25 |
| 24-pin | pcb | .35 | ww | 1.10 |
| 28-pin | pcb | .35 | ww | 1.45 |
| 40-pin | pcb | .50 | ww | 1.25 |

| | | | |
|---|---|---|---|
| Molex pins | .01 | To-3 Sockets | .45 |
| 2 Amp Bridge | | 100-prv | 1.20 |
| 25 Amp Bridge | | 200-prv | 1.95 |

## TRANSISTORS, LEDS, etc.

| | | |
|---|---|---|
| 2N2222A | NPN (2N2222 Plastic .10) | .15 |
| 2N2907A | PNP | .15 |
| 2N3906 | PNP (Plastic) | .10 |
| 2N3904 | NPN (Plastic) | .10 |
| 2N3054 | NPN | .35 |
| 2N3055 | NPN 15A 60v | .50 |
| T1P125 | PNP Darlington | .35 |
| LED Green, Red, Clear, Yellow | | .15 |
| D.L.747 | 7 seg 5/8" High com-anode | 1.95 |
| XAN72 | 7 seg com-anode (Red) | 1.25 |
| MAN71 | 7 seg com-anode (Red) | 1.25 |
| MAN3610 | 7 seg com-anode (Orange) | 1.25 |
| MAN82A | 7 seg com-anode (Yellow) | 1.25 |
| MAN74A | 7 seg com-cathode (Red) | 1.50 |
| FND359 | 7 seg com-cathode (Red) | 1.25 |

## C MOS

| | |
|---|---|
| 4000 | .15 |
| 4001 | .15 |
| 4002 | .20 |
| 4004 | 3.95 |
| 4006 | .95 |
| 4007 | .35 |
| 4008 | .95 |
| 4009 | .45 |
| 4010 | .45 |
| 4011 | .20 |
| 4012 | .20 |
| 4013 | .40 |
| 4014 | .95 |
| 4015 | .90 |
| 4016 | .35 |
| 4017 | 1.10 |
| 4018 | 1.10 |
| 4019 | .50 |
| 4020 | .85 |
| 4021 | 1.00 |
| 4022 | .85 |
| 4023 | .25 |
| 4024 | .75 |
| 4025 | .30 |
| 4026 | 1.95 |
| 4027 | .50 |
| 4028 | .95 |
| 4030 | .35 |
| 4033 | 1.50 |
| 4034 | 2.45 |
| 4035 | 1.25 |
| 4040 | 1.35 |
| 4041 | .69 |
| 4042 | .95 |
| 4043 | .95 |
| 4044 | .95 |
| 4046 | 1.75 |
| 4049 | .45 |
| 4050 | .45 |
| 4066 | .95 |
| 4069 | .40 |
| 4071 | .35 |
| 4081 | .70 |
| 4082 | .45 |
| MC14409 | 14.50 |
| MC14419 | 4.85 |

## — T T L —

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 7400 | .15 | 7473 | .25 | 74176 | 1.25 | 74H72 | .45 | 74S133 | .40 | | |
| 7401 | .15 | 7474 | .30 | 74180 | .75 | 74H101 | .75 | 74S140 | .55 | | |
| 7402 | .20 | 7475 | .35 | 74181 | 2.25 | 74H103 | .75 | 74S151 | .30 | | |
| 7403 | .20 | 7476 | .40 | 74182 | .95 | 74H106 | .95 | 74S153 | .35 | | |
| 7404 | .15 | 7480 | .55 | 74190 | 1.75 | | | 74S157 | .75 | | |
| 7405 | .25 | 7481 | .75 | 74191 | 1.05 | 74L00 | .25 | 74S158 | .30 | | |
| 7406 | .35 | 7483 | .95 | 74192 | .75 | 74L02 | .25 | 74S194 | 1.05 | | |
| 7407 | .55 | 7485 | .75 | 74193 | .85 | 74L03 | .30 | 74S257 (8123) | 1.05 | | |
| 7408 | .25 | 7486 | .25 | 74194 | 1.25 | 74L04 | .30 | | | | |
| 7409 | .15 | 7489 | 1.35 | 74195 | .95 | 74L10 | .30 | 74LS00 | .25 | | |
| 7410 | .10 | 7490 | .55 | 74196 | 1.25 | 74L20 | .35 | 74LS01 | .35 | | |
| 7411 | .25 | 7491 | .95 | 74197 | 1.25 | 74L30 | .45 | 74LS02 | .35 | | |
| 7412 | .30 | 7492 | .95 | 74198 | 2.35 | 74L47 | 1.95 | 74LS04 | .30 | | |
| 7413 | .35 | 7493 | .35 | 74221 | 1.00 | 74L51 | .45 | 74LS05 | .45 | | |
| 7414 | 1.10 | 7494 | .75 | 74367 | .85 | 74L55 | .65 | 74LS08 | .25 | | |
| 7416 | .25 | 7495 | .60 | | | 74L72 | .45 | 74LS09 | .35 | | |
| 7417 | .40 | 7496 | .80 | 75108A | .35 | 74L73 | .40 | 74LS10 | .35 | | |
| 7420 | .15 | 74100 | 1.15 | 75110 | .35 | 74L74 | .45 | 74LS11 | .35 | | |
| 7426 | .30 | 74107 | .35 | 75491 | .50 | 74L75 | .55 | 74LS20 | .25 | | |
| 7427 | .45 | 74121 | .35 | 75492 | .50 | 74L93 | .55 | 74LS21 | .25 | | |
| 7430 | .15 | 74122 | .55 | | | 74L123 | .85 | 74LS22 | .25 | | |
| 7432 | .30 | 74123 | .55 | 74H00 | .15 | | | 74LS32 | .40 | | |
| 7437 | .30 | 74125 | .45 | 74H01 | .25 | 74S00 | .35 | 74LS37 | .35 | | |
| 7438 | .35 | 74126 | .35 | 74H04 | .20 | 74S02 | .35 | 74LS40 | .45 | | |
| 7440 | .25 | 74132 | 1.35 | 74H05 | .20 | 74S03 | .30 | 74LS42 | 1.10 | | |
| 7441 | 1.15 | 74141 | .90 | 74H08 | .35 | 74S04 | .30 | 74LS51 | .50 | | |
| 7442 | .45 | 74150 | .85 | 74H10 | .35 | 74S05 | .35 | 74LS74 | .65 | | |
| 7443 | .65 | 74151 | .65 | 74H11 | .35 | 74S08 | .35 | 74LS86 | .65 | | |
| 7444 | .45 | 74153 | .75 | 74H15 | .45 | 74S10 | .35 | 74LS90 | .95 | | |
| 7445 | .65 | 74154 | .95 | 74H20 | .30 | 74S11 | .35 | 74LS93 | .95 | | |
| 7446 | .95 | 74156 | .95 | 74H21 | .25 | 74S20 | .35 | 74LS107 | .85 | | |
| 7447 | .95 | 74157 | .65 | 74H22 | .40 | 74S40 | .20 | 74LS123 | 1.00 | | |
| 7448 | .65 | 74161 | .85 | 74H30 | .20 | 74S50 | .20 | 74LS151 | .95 | | |
| 7450 | .25 | 74163 | .85 | 74H40 | .25 | 74S51 | .25 | 74LS153 | 1.20 | | |
| 7451 | .25 | 74164 | .60 | 74H50 | .25 | 74S64 | .20 | 74LS157 | .85 | | |
| 7453 | .20 | 74165 | 1.50 | 74H51 | .25 | 74S74 | .35 | 74LS164 | 1.90 | | |
| 7454 | .25 | 74166 | 1.35 | 74H52 | .15 | 74S112 | .60 | 74LS367 | .75 | | |
| 7460 | .40 | 74175 | .80 | 74H53J | .25 | 74S114 | .65 | 74LS368 | .75 | | |
| 7470 | .45 | | | 74H55 | .20 | | | 74C04 | .25 | | |
| 7472 | .40 | | | | | | | 74C151 | 2.25 | | |

## 9000 SERIES

| | | | |
|---|---|---|---|
| 9301 | .85 | 95H03 | 1.10 |
| 9309 | .35 | 9601 | .45 |
| 9322 | .75 | 9602 | .45 |

## MICRO'S, RAMS, CPU'S, ETC.

| | |
|---|---|
| 74S188 | 3.00 |
| 1702A | 4.50 |
| MM5314 | 3.00 |
| MM5316 | 3.50 |
| 2102-1 | 1.45 |
| 2102L-1 | 1.75 |
| TR1602B | 4.50 |
| TMS 4044-45NL | 14.50 |
| 8080AD | 12.00 |
| 8T13 | 1.50 |
| 8T23 | 1.50 |
| 8T24 | 2.00 |
| 8T97 | 1.00 |
| 2107B-4, A | 4.00 |
| 2708 | 11.50 |

## LINEARS, REGULATORS, etc.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MCT2 | .95 | LM320T5 | 1.65 | LM340K15 | 1.25 | LM723 | .50 |
| 8038 | 3.95 | LM320T12 | 1.65 | LM340K18 | 1.25 | LM725N | 2.50 |
| LM201 | .75 | LM320T15 | 1.65 | LM340K24 | .95 | LM739 | 1.50 |
| LM301 | .45 | LM324N | .95 | 78L05 | .75 | LM741 (8-14) | .25 |
| LM308 (Mini) | .95 | LM339 | .95 | 78L12 | .75 | LM747 | 1.10 |
| LM309H | .65 | 7805 (340T5) | .95 | 78L15 | .75 | LM1307 | 1.25 |
| LM309K (340K-5) | .85 | LM340T12 | 1.00 | 78M05 | .75 | LM1458 | .95 |
| LM310 | 1.15 | LM340T15 | 1.00 | LM373 | 2.95 | LM3900 | .50 |
| LM311D (Mini) | .75 | LM340T18 | 1.00 | LM380 (8-14 PIN) | .95 | LM75451 | .65 |
| LM318 (Mini) | .95 | LM340T24 | .95 | LM709 (8, 14 PIN) | .25 | NE555 | .50 |
| LM320K5 (7905) | 1.65 | LM340K12 | 1.65 | LM711 | .45 | NE556 | .95 |
| LM320K12 | 1.65 | | | | | NE565 | .95 |
| | | | | | | NE566 | 1.75 |
| | | | | | | NE567 | 1.35 |

# INTEGRATED CIRCUITS UNLIMITED 19

**7889 Clairemont Mesa Boulevard, San Diego, California 92111**
**(714) 278-4394** *(Calif. Res.)*

All orders shipped prepaid — No minimum
Open accounts invited — COD orders accepted

Discounts available at OEM Quantities — California Residents add 6% Sales Tax
All IC's Prime/Guaranteed. All orders shipped same day received.

### SPECIAL DISCOUNTS

| Total Order | Deduct |
|---|---|
| $35 - $99 | 5% |
| $100 - $300 | 10% |
| $301 - $1000 | 15% |
| $1000 - Up | 20% |

**24 Hour Toll Free Phone 1-800-854-2211**

**American Express / BankAmericard / Visa / MasterCharge**

# S.D. COMPUTER PRODUCTS AN EMPIRE IND. CO.

P.O. BOX 28810K DALLAS, TEXAS 75228 S2

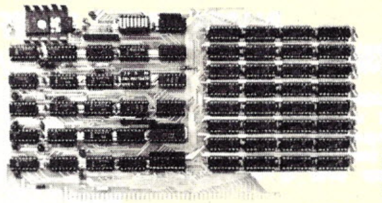
## 64K FOR $995.00

At last! The popular Expandoram is available in a 16k multiple version. Similar to our 32k Expandoram, the new Super Expandoram is offered in 16k, 32k, 48k and 64k. Low power devices mean the very lowest power consumption. Allow 3-4 weeks for delivery.

| | |
|---|---|
| 16K — $281.00 | 48K — $757.00 |
| 32K — $519.00 | 64K — $995.00 |

## 32K FOR $475 EXPANDORAM KIT 24K FOR $367.00

MEMORY CAPACITY
MEMORY ADDRESSING
MEMORY WRITE PROTECTION
8K, 16K, 24K, 32K using Mostek MK4115 with 8K boundaries and protection. Utilizes DIP switches. PC board comes with sockets for 32K operation. Orders now being accepted. Allow 6 to 8 weeks for delivery.



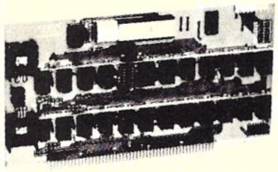
INTERFACE CAPABILITY
Control, data and address inputs utilizes low power Schottky devices.
POWER REQUIREMENTS
+8VDC 400MA DC
+18VDC 400MA DC
−18VDC 30MA DC
on board regulation is provided. On board (invisible) refresh is provided with no wait states or cycle stealing required.
MEMORY ACCESS TIME IS 375ns.
Memory Cycle Time is 500ns.

Buy an S100 compatible 8K Ram Board and upgrade the same board to a maximum of 32K in steps of 8K at your option by merely purchasing more ram chips from S.D. Sales! At a guaranteed price — Look at the features we have built into the board.

16K FOR $259.00 8K FOR $151.00

## Z-80 CPU BOARD KIT — $139.

CHECK THE ADVANCED FEATURES OF OUR Z—80 CPU BOARD: Expanded set of 158 instructions, 8080A software capability, operation from a single 5VDC power supply; always stops on an M1 state, true sync generated on card (a real plus feature!), dynamic refresh and NMI available, either 2MHZ or 4MHZ operation, quality double sided plated through PC board; parts plus sockets priced for all IC's. *Add $10 extra for Z—80A chip which allows 4MHZ operation. **Z—80 chip with Manual** — 29.95



## S.D. SALES NEW EXPANDABLE EPROM BOARD

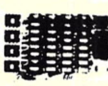
16K or 32K EPROM $49.95 w/out EPROM
Allows you to use either 2708's for 16K of Eprom or 2716's for 32K of Eprom.
KIT FEATURES:

1. All address lines & data lines buffered.
2. Quality plated through P.C. Board, including solder mask and silk screen.
3. Selectable wait states.
4. On board regulation provided.
5. All sockets provided w/board.

WE CAN SUPPLY 450ns 2708's AT $11.95 WHEN PURCHASED WITH BOARD.

## 4K LOW POWER RAM KIT

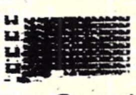
Fully Buffered — on board regulated — reduced power consumption utilizing low power 21L02 — 1 500ns RAMS — Sockets provided for all IC's. Quality plated through PC board. *Add $10. for 250ns RAM operation.



**The Whole Works - $79.95**

## 8K LOW POWER RAM — $159.95

Fully assembled and tested. Not a kit. Imsai — Altair — S-100 Buss compatible, uses low power static 21L02-500ns fully buffered on board regulated, quality plated through PC board, including solder mask. 8 pos. dip switches for address select.



**250 ns Operation $189.95**

## 6 DIGIT ALARM CLOCK KIT

Features: Litronix dual 1/2" displays, Mostek 50250 super clock chip, single I.C. segment driver, SCR digit drivers. Kit includes all necessary parts (except case). Xfmr optional. Eliminate the hassle.
AC XFMR — $1.50 Case $3.50 **$12.95**

## Bowmar 4 Digit LED Readout Array Full ½" Litronix Jumbo Dual Digit LED Displays

4 JUMBO .50" DIGITS ON ONE STICK! WITH COLONS & AM/PM INDICATOR
$3.95

| | |
|---|---|
| DL 722 - C.C. | DL 728 - C.C. |
| DL 721 8 C.A. | DL 727 - C.A. |
| 99c | $1.29 |

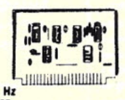
## NEW FROM S.D. "VERSAFLOPPY"™ KIT

THE VERSATILE FLOPPY DISK CONTROLLER
**ONLY $149.00**

Features: IBM 3740 Soft Sectored Compatible. S-100 BNS Compatible for Z-80 or 8080. Controls up to 4 Drives (single or double sided). Directly controls the following drives:
1. Shugart SA400/450 Mini Floppy.
2. Shugart SA800/850 Standard Floppy.
3. PERSCI 70 and 277.
4. MFE 700/750.
5. CDC 9404/9406.
34 Pin Connector for Mini Floppy. 50 Pin Connector for Standard Floppy. Operates with modified CP/M operating system and C-Basic Compiler.
The new "Versafloppy" from S.D. Computer Products provides complete control for many of the available Floppy Disk Drives, Both Mini and Full Size. At the heart of "Versafloppy" is the powerful Western Digital FD1771B-1 Single Density Controller Chip. This allows a great flexibility via Control Software. Listings for Control Software are ncluded in the price.

FD 1771B-1 CHIP ALONE $39.95

## Low Cost Cassette Interface Kit



Features: Play and record K.C. Standard 2400/1200 Hz tapes, 300 Baud, TTL I/O Compatible, Phase Lock Loop, Both 22 Pin Connector and 8 Pin Molex Connector. Comes partially assembled. Oscillator and phase lock loop pre-tuned to K.C. Standard. Selector switch sends cassette data or auxiliary input data to microprocessor. LED indicates logic 1 level.
**$19.95**

## Jumbo LED Car Clock Kit

FEATURES:
A. Bowmar Jumbo .5 inch LED array.
B. MOSTEK — 50250 — Super clock chip.
C. On board precision crystal time base.
D. 12 or 24 hour Real Time format.
E. Perfect for cars, boats, vans, etc.
F. PC board and all parts (less case) inc.
Alarm option — $1.50
AC XFMR — $1.50

**$16.95**

## RAMS

| | |
|---|---|
| 21L02 - 500NS | 8/11 50 |
| 21L02 - 250NS | 8/15 95 |
| 2114 — 4K | 14 95 |
| 1101A — 256 | 8/$4.00 |
| 1103 — 1K | 35 |
| MK 4115 - 8K | 19 45 |
| 74S 200 - 256 | 3 95 |

## CPU's

| | |
|---|---|
| Z—80 includes manual | 29 95 |
| Z—80A includes manual | 34 95 |
| 8080A CPU 8 BIT | 11 95 |
| 8008 CPU 8 BIT | 6 95 |

## PROMS

| | |
|---|---|
| 1702A - 1K - 1.5us | 3.95 or 10/35. |
| 2708 - 8K - 450ns | 14.95 |
| 5204 - 4K | 7.95 |
| 82S129 — 1K | 2.50 |
| 2708U 8K signetics 650ns | 9.95 |

## COUNTER CHIPS

| | |
|---|---|
| MK50397 6 Digit elapsed timer | 8.95 |
| MK50250 Alarm clock | 4.99 |
| MK50380 Alarm chip | 2.95 |
| MK50395 6 digit up/dn. count. | 12.95 |
| MK5002 4 digit counter | 8.95 |
| MK5021-Cal. chip sq. root | 2.50 |

## ★ ★ ★ ★SUPER FLOPPY SPECIAL★ ★ ★ ★

S. D. SALES,' VERSAFLOPPY S-100 CONTROLLER BOARD PLUS SHUGART SA 400 FLOPPY DISK DRIVE INCLUDING CABLE FOR ONLY
**$479.00**

## MICRO-DIP $1.95

New — Series 2300
The World's Smallest Coded BCD Dual-In-Line Switch! PC Mount
2300 02G BCD 1-2-4-8
2300 12G BCD 1-2-4-8
Compliment

## ★ ★ JOY STICKS★ ★

FOUR 100 K-OHMS POTS
Ideal for electronic games

$3.95

## Z-80 Programming Manual

IN DEPTH DETAIL OF THE Z-80 CPU MICRO-COMPUTER
S. D. SALES SPECIAL
**$9.95**

| | |
|---|---|
| Thermistors 1.5K ohm | 5/$1.00 |
| Tantalum Caps 1 mfd. 20VOC P.C. Leads | 15/$1.00 |
| Flat Pack IC Assort. | 20/$1.00 |
| Electrical Coil 13T Type C - 10T Type C | 12/$1.00 |
| 2 Transistor Audio | 8/$1.00 |
| Trimmer Pots 10K, 20K, 25K, Mini | 10/$1.00 |
| Disc Caps For Bypass .01 mfg - 100 WUDC PC Leads | 40/$1.00 |
| New Cambion Jacks Part #450-4352 Gold Plated | 50/$1.00 |

| | |
|---|---|
| Silicon Rectifier Special 1N4007. 1 amp 1000 PN. | 10/$1.00 |
| Photocell Assortment | 12/$1.00 |
| Plastic Readout Filters Amber | 6/$1.00 |
| Disc Cap Assortment | 60/$1.00 |
| P.C. Lead Diodes 1N4148 1N 914 | 100/$2.00 |
| 1N4002-1A-100 PN | 40/$1.00 |
| MICA Trimmer PC402 Miniature 1.5-20 P.C. P.C. Mount | 4/$1.00 |
| Resistor Special 22 ohm Carbon Comp. | 25/$1.00 |
| Resistor Assortment 1/4 W 5% & 10% PC leads | 200/$1.50 |

CHOOSE $1. FREE MERCHANDISE FROM ASTERISK ITEMS ON EACH $15 ORDER

## MICROPROCESSOR CHIPS

| | |
|---|---|
| 8212 - I/O port | 3.50 |
| 8214 — P.I.C. | 12.95 |
| 8216 — Non Invert Bus | 4.95 |
| 8224 — Clock Gen. | 4.95 |
| 8226 — Invert Bus | 3.95 |
| PIO for Z—80 | 14.95 |
| CTC for Z—80 | 14.95 |
| 8228 Sys. Controller | 8.20 |
| 8251 Prog. comm. interfact. | 10.95 |
| 8255 prog. prep. interface | 13.50 |
| 8820 Dual Line Recr | 1.75 |
| 8830 Dual Line Dr. | 1.75 |
| 2513 Char. Gen. | 7.50 |
| 8838 Quad Bus. Recvr. | 2.00 |
| 74LS138N — 1/8 decoder | .99 |
| 8T97-Hex Tri-State Buffer | 1.25 |
| 1488/1489 RS232 | 1.50 |
| TR 1602B Uart | 3.95 |
| TR 1863 Uart | 8.50 |
| FD 1771B-1 | 39.95 |

## CMOS

| | | | |
|---|---|---|---|
| 4001 | 19 | 4029 | 99 |
| 4002 | 19 | 4042 | 69 |
| 4011 | 19 | 4047 | 1.50 |
| 4013 | 32 | 4049 | 35 |
| 4016 | 32 | 4069 | 23 |
| 4017 | 95 | 4071 | 19 |
| 4020 | 97 | 4076 | 97 |
| 4022 | 97 | 14518 | 1.10 |
| 4024 | 75 | 14528 | 85 |
| 4027 | 39 | 14529 | 85 |

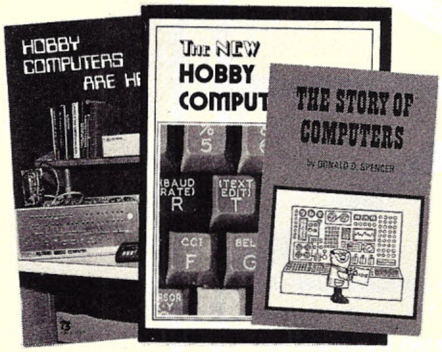
CALL IN YOUR BANKAMERICARD (VISA) OR MASTER CHARGE ORDER IN ON OUR CONTINENTAL TOLL FREE WATTS LINE:
**1—800—527—3460**

Texas Residents Call Collect:
**214/271—0022**
DEALER INQUIRIES INVITED!

NO COD's. TEXAS RESIDENTS ADD 5% SALES TAX. ADD 5% OF ORDER FOR POSTAGE & HANDLING. ORDERS UNDER $10. ADD 75c HANDLING. FOREIGN ORDERS — U. S. FUNDS ONLY!

KB book nook KB book nook

## beginner's · introductory






●**THE STORY OF COMPUTERS** by Donald D. Spencer is to computer books what *Dick and Jane* is to novels . . . extremely elementary, gives the non-computerist a fair idea of what the hobbyist is talking about when he speaks computer lingo. Attempts to explain what computers are and can do to a spouse, child or any un-electronics-minded friend. $4.95.*

●**HOBBY COMPUTERS ARE HERE** If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the circuits, the basics of programming, along with a couple of TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid for newcomers. $4.95.*

●**THE NEW HOBBY COMPUTERS!** This book takes it from where "Hobby Computers Are Here" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards, a Baudot monitor/editor system, an audible logic probe for finding those tough problems, a ham's computer, a computer QSO machine . . . and much, much more! $4.95.*

●**HOME COMPUTERS: $2^{10}$ Questions & Answers** by Rich Didday. Two books aimed exclusively at the novice computer hobbyist/home computer user. Written in a rather unusual style which has a beginner asking questions which are answered by a person with a substantial background in computers and personal computing. The questions are just the kind beginners come up with . . . and the answers are presented in easy-to-understand terms (usually with a diagram to illustrate the point). Both the hardware and software aspects of home computing are covered from A to Z. An index in both books makes them ideal as reference material *for anyone.* Volume I: Hardware — $7.95*; Volume 2: Software — $6.95*.

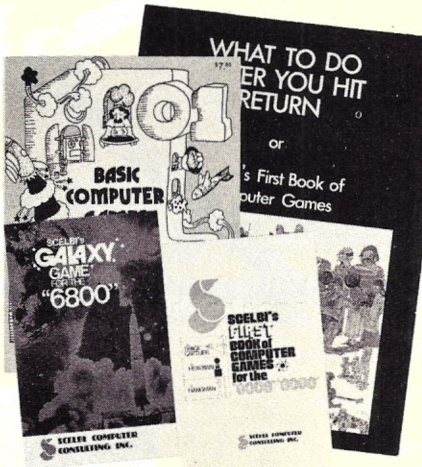
●**MICROPROCESSORS FROM CHIPS TO SYSTEMS** by Rodnay Zaks is a complete and detailed introduction to microprocessors and microcomputer systems. No preliminary knowledge of computers or microprocessors is required to read this book, although a basic engineering knowledge is naturally an advantage. Intended for all wishing to understand the concepts, techniques and components of microprocessors in a short time. $9.95.*

●**MICROCOMPUTER PRIMER** by Mitchell Waite and Michael Pardee. Describes basic computer theory, explains numbering systems, and introduces the reader to the world of programming. Describes the world of microcomputing in "real world" terminology. No better way of getting involved with the exciting new hobby of microcomputing. $7.95.*

● **INTRODUCTION TO MICROPROCESSORS** by Charles Rockwell of MICROLOG is an ideal reference for the individual desiring to understand the hardware aspects of microprocessor systems. Describes the hardware details of computer devices in terms the beginner can understand, instead of treating the micro chip as a "black box." General information about hardware systems is provided. Specific systems are not described and programming is only briefly discussed. $17.50 US and Canada, $20 elsewhere.*

●**AN INTRODUCTION TO MICROCOMPUTERS, VOLS. 1 AND 2** by Adam Osborne Associates, are references dealing with microcomputer architecture in general and specifically with details about most of the common chips. These books are not software-oriented, but are invaluable for the hobbyist who is into building his own interfaces and processors. Volume 1 is dedicated to general hardware theory related to micros, and Volume 2 discusses the practical details of each micro chip. (Detailed review in Kilobaud #2) Volume 1 — $7.50*; Volume II — $12.50.*

## computer games




●**WHAT TO DO AFTER YOU HIT RETURN** PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, muchly illustrated. Lunar landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $8.00.*

●**SCELBI'S GALAXY GAME FOR THE "6800"** Here's a new twist in computer games by Robert Findley/Raymond Edwards. "Galaxy" pits the operator of a spaceship against alien craft, as well as such variables as speed, time, and ammunition. No two games are the same! $14.95.*

● **101 BASIC COMPUTER GAMES** Okay, so once you get your computer up and running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you, from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50.*

●**SCELBI'S FIRST BOOK OF COMPUTER GAMES** Need a game for your 8008 or 8080 microprocessor? Has three popular games, "Space Capture," "Hexpawn," and "Hangman." Complete flowcharts, logic description, program listing, and instructions are provided. A must for the game freak! $14.95.*

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Book Department    Peterborough NH 03458
Be sure to include check or detailed credit card information

* *Add $1.00 shipping & handling charge for each order.* Note: Prices subject to change on books not published by 73 Magazine.

## software•programming

●**DISCOVERING BASIC – A Problem Solving Approach** by Robert E. Smith deals with progressively more complex problems which allow the reader to discover the vocabulary of BASIC language as he develops skill and confidence in putting it to work. Clear and concise explanations. Problems used cover a wide range of interests – insurance, geometry, puzzles, economics, etc. $6.85.*

● **BASIC New 2nd Edition.** by Bob Albrecht. Self-teaching guide to the computer language you will need to know for use with your microcomputer. This is one of the easiest ways to learn computer programming. $4.95.*

●**A QUICK LOOK AT BASIC** by Donald D. Spencer. A perfect reference for the beginning programmer. Assumes that the reader has no previous programming experience and is a self-teaching guide for the individual desiring to learn the fundamentals of BASIC. $4.95.*

●**MY COMPUTER LIKES ME . . . WHEN I SPEAK BASIC** An introduction to BASIC . . . simple enough for your kids. If you want to teach BASIC to anyone quickly, this book is the way to go. $2.00.*

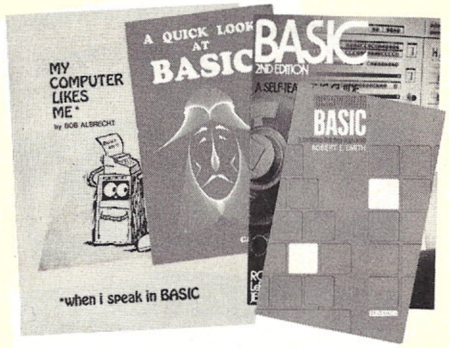
●**FUN WITH COMPUTERS AND BASIC** by Donald D. Spencer, contains an easy-to-understand explanation of the BASIC Programming Language and is intended for persons who have had no previous exposure to computer programming. Over half the book is devoted to problems using games, puzzles, and mathematical recreations. A superior book for self-teaching and learning computer programming. $6.95.*

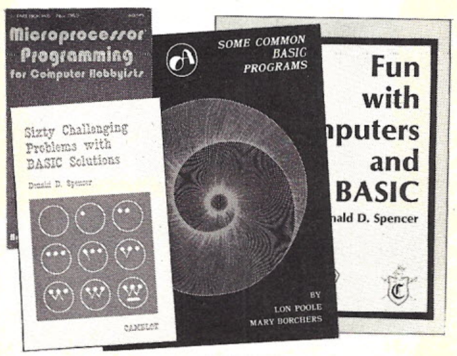
● **SIXTY CHALLENGING PROBLEMS WITH BASIC SOLUTIONS** by Donald Spencer, provides the serious student of BASIC programming with interesting problems and solutions. No knowledge of math above algebra required. Incudes a number of game programs, as well as programs for financial interest, conversions and numeric manipulations. $6.95.*

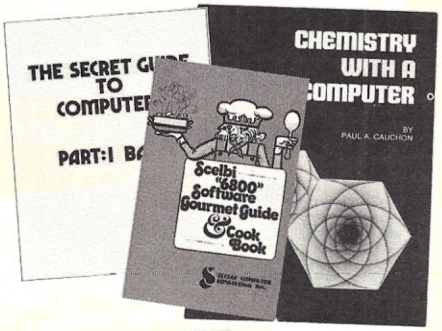
●**THE SECRET GUIDE TO COMPUTERS** Parts 1, 2, and 3 by Russ Walter. **Part One** describes computers in general, and after reading for ten minutes you will be writing simple BASIC programs! **Part Two** discusses computer applications. It's one thing to master the syntax of a language such as BASIC and another to solve problems using the new tool. **Part Three** describes programming languages. Ever heard of APL and QLISP? BASIC is not the only language used to program computers. 7th Edition. Part I - $2.75*; Part II – $2.50*; Part III – $3.50.*

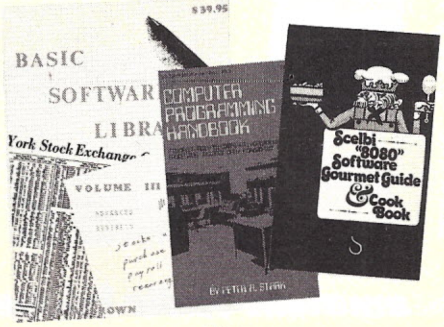
●**SOME COMMON BASIC PROGRAMS** published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the standalone mode. $7.50 paperback.*

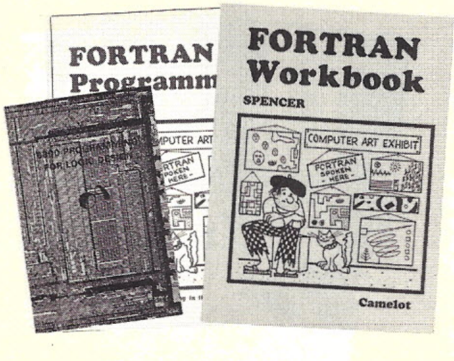
●**Scientific Research Instruments' BASIC SOFTWARE LIBRARY** is a complete do-it-yourself kit. Written in everybody's BASIC immediately executable in ANY computer with at least 4K, no other peripherals needed. Vol. I contains business and recreational programs and is 300 pages. Vol. II is 260 pages and contains math, engineering, statistics and plotting programs. Vol. III contains money managing, advanced business programs such as billing, A/R, inventory, payroll, etc. Vol. IV contains general purpose programs like loans, rates, retirement, plus games: Poker, Enterprise (take charge while Capt. Kirk is away), Football and more! Vol. V is filled with experimenter's programs including games, pictures and misc. problems like "logic." Vols. I & II $24.95,* Vol. III $39.95,* Vol. IV & V $9.95 each.*




●**MICROPROCESSOR PROGRAMMING for Computer Hobbyists** by Neill Graham is for the hobbyist interested in intermediate and advanced techniques of programming and data structuring. Written to take up where the computer manufacturers' instruction manuals and the introductory programming language texts leave off. $8.95.*

● **ADVANCED BASIC** – Applications and Problems by James Coan is for those who want to extend their expertise with BASIC. Offers advanced techniques and applications. $6.95.*

● **8080 PROGRAMMING FOR LOGIC DESIGN** Ideal reference for an in-depth understanding of the 8080 processor. Application-oriented and the 8080 is discussed in light of replacing conventional, hard-wired logic. Practical design considerations are provided for the implementation of an 8080-based control system. $7.50.*

●**8080 SOFTWARE GOURMET GUIDE AND COOKBOOK** If you have been spending too much time developing simple routines for your 8080, try this new book by Scelbi Computing and Robert Findley. Describes sorting, searching, and many other routines for the 8080 user. $9.95.*

● **6800 PROGRAMMING FOR LOGIC DESIGN** Oriented toward the industrial user, this book describes the process by which conventional logic can be replaced by a 6800 microprocessor. Provides practical information that allows an experimenter to design a complete micro control system for the "ground up." $7.50.*

●**6800 SOFTWARE GOURMET GUIDE & COOKBOOK** If you have been spending too much time developing routines for your 6800 microprocessor, try the new book by Scelbi Computing and Robert Findley. Describes sorting, searching, and many other routines for the 6800 user. $9.95.*

●**FORTRAN PROGRAMMING** by Donald Spencer. FORTRAN was designed for complex numeric calculations; and possesses extended I/O capability. It is easily learned, as it is an English-like computer language. $7.50.*

● **FORTRAN WORKBOOK** by Donald Spencer. Provides practical examples and problems to solve. Flowcharting is also discussed. Almost all micros support BASIC – it won't be long before FORTRAN is commonplace. $3.95.*

●**CHEMISTRY WITH A COMPUTER** by Paul A. Cauchon, contains a collection of tutorial, simulation and problem-generation computer programs. Usable with almost any chemistry course in the high school or college level. $9.95.*

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Book Department Peterborough NH 03458
Be sure to include check or detailed credit card information

✱ *Add $1.00 shipping & handling charge for each order.* Note: Prices subject to change on books not published by 73 Magazine.

KB book nook KB book nook

KB book nook

## hardware

**• MICROPROCESSOR INTERFACING TECHNIQUES** by Austin Lesea & Rodnay Zaks will teach you how to interconnect a complete system and interface it to all the usual peripherals. It covers hardware and software skills and techniques, including the use and design of model buses such as the IEEE 488 or S100. $9.95.*

**• TTL COOKBOOK** by Donald Lancaster. Explains what TTL is, how it works, and how to use it. Discusses practical applications, such as a digital counter and display system, events counter, electronic stopwatch, digital voltmeter, and a digital tachometer. $8.95.*

**• CMOS COOKBOOK** by Don Lancaster. Details the application of CMOS, the low power logic family suitable for most applications presently dominated by TTL. Required reading for every serious digital experimenter! $9.95.*

**• TVT COOKBOOK** by Don Lancaster, describes the use of a standard television receiver as a microprocessor CRT terminal. Explains and describes character generation, cursor control and interface information in typical, easy-to-understand Lancaster style. $9.95.*

**• BUILD-IT BOOK OF DIGITAL ELECTRONIC TIMEPIECES** by Robert Haviland is a data-packed guide to building every timekeeping device you can imagine: rugged shipboard clocks, second-splitting digital IC chronometers, decorator digital clocks, a precision timer, a frequency-period meter, a tide and moon clock, an automatic alarm setter, etc. Including full-size printed circuit board layouts. $6.95.*


**• THE "COMPULATOR" BOOK – Building Super Calculators & Minicomputer Hardware with Calculator Chips** by R. P. Haviland, provides ideas, design info and printed circuit boards for calculator chip projects, measure time, tie in with a Teletype to create a virtually infinite memory system, and countless other functions. $7.95.*

### TEST EQUIPMENT LIBRARY

**• VOL I COMPONENT TESTERS** Build your own test equipment and save a bundle (and have a lot of fun). Volume I of the 73 Test Equipment Library shows you how to build and use transistor testers (8 of 'em), three diodes testers, 3 IC testers, 9 voltmeters and VTVMs, 8 ohmmeter, 3 inductance meters, and a raft of other gadgets for checking temperature, crystals, Q, etc. $4.95.*

**• VOL II AUDIO FREQUENCY TESTERS** If you're into audio such as digital cassette recording, RTTY, Baudot vs ASCII, SSTV, SSB, Touchtone or even hi-fi you'll want to have this book full of home built test equipment projects. Volume II $4.95.*

**• VOL III RADIO FREQUENCY TESTERS** This is of more interest to hams and CBers: test equipment you can build for checking out transmitters and receivers, signal generators, noise generators, crystal calibrators, GDOs, dummy loads. $4.95.*

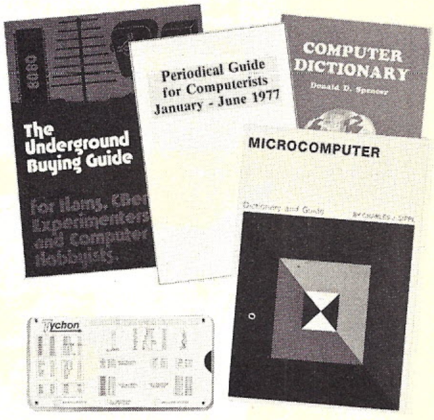
**• VOL. IV IC TEST EQUIPMENT** Become a trouble-shooting wizard. All you need to know about pulse, audio and sync generators, frequency counters, digital component testers, logic probes and more! Plus a cumulative index for all four volumes of the 73 Test Equipment Library. $4.95.*

## general

**• COMPUTER DICTIONARY** by Donald D. Spencer. Defines words and acronyms used by computerists in a clear, easy to understand style. This reference is a must for the individual getting started in the world of microcomputers. $5.95.*

**• BRAND NEW DICTIONARY** This microcomputer dictionary fills the need to become quickly acquainted with the terminology and nomenclature of the revolution in computers. There is also a comprehensive electronics/computer abbreviations and acronyms section. $15.95.*

**• THE UNDERGROUND BUYING GUIDE** Here is a handy guide for the electronics enthusiast. Over 600 sources of equipment and literature are provided. Cross-referenced for ease of use. Electronic publishing houses are also listed. $5.95 each.*




**• 1976 PERIODICAL GUIDE FOR COMPUTERISTS** is a 20-page book which indexes over 1,000 personal computing articles for the entire year of 1976 from Byte, Creative Computing, Digital Design, Dr. Dobbs Journal, EDN, Electronic Design, Electronics, Interface Age, Microtrek, Peoples Computer Company, Popular Electronics, QST, Radio Electronics, SCCS Interface and 73 Amateur Radio. Price $3.00.* New January – June 1977 Edition (includes Kilobaud) – $3.00.*

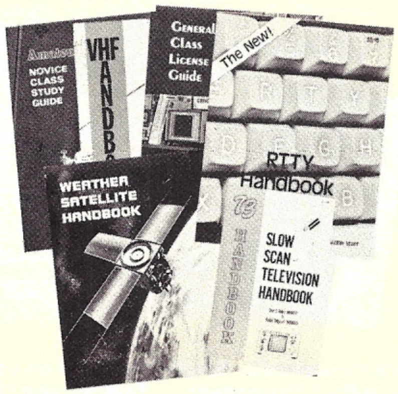
**• TYCHON'S 8080 OCTAL CODE CARD** Slide rule-like aid for programming and debugging 8080 software contains all the mnemonics and corresponding octal codes. Also available, Tychon's *8080 Hex Code Card*, same as above only has hex codes instead of octal. $3.00 each.*

## amateur radio books

**• NOVICE STUDY GUIDE** The most complete Novice study guide available. It is brand new. This is not only invaluable for anyone wanting to get started in amateur radio, but also it is about the only really simple book on the fundamentals of electricity and electronics. $4.95.*

**• GENERAL CLASS STUDY GUIDE** Takes over on theory where the Novice book leaves off. You'll need to know the electronic theory in this to work with computers and you'll not find an easier place to get the information. $5.95.*

**• SSTV HANDBOOK** This excellent book tells all about it, from its history and basics to the present state-of-the-art techniques. Hardbound $7,* Softbound $5.*



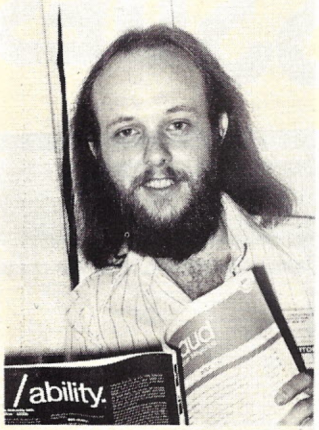

**• VHF ANTENNA HANDBOOK** This new handbook details the theory, design and construction of hundreds of different VHF and UHF antennas. Packed with fabulous antenna projects you can build. $4.95.*

**• WEATHER SATELLITE HANDBOOK** Simple equipment and methods for getting good pictures from the weather satellite. Dr. Taggart WB8DQT $4.95.*

**• THE NEW RTTY HANDBOOK** is a brand new 1977 edition and the only up-to-date RTTY book available. The state-of-the-art has been changing radically and has made all previous RTTY books obsolete. It has the latest circuits, great for the newcomer and the expert alike. $5.95.*

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to:
Kilobaud Book Department Peterborough NH 03458 Be sure to include check or detailed credit card information

** Add $1.00 shipping & handling charge for each order.* Note: Prices subject to change on books not published by 73 Magazine.




Dave Shirk looks up from his copy of KB.

### WHO'S BEHIND THE KILOBAUD?

In between selling subscriptions and seeing the other exhibits at computer shows, Wayne snaps pictures of people you have been seeing and will be seeing at shows. How many of them can you recognize behind the Kilobauds?

On page 143 of the March issue there was a picture of Dave Shirk, President of Technical Systems Consultants. Okay . . . so you missed that one. Now that you know who he is, be sure to get together with Dave at the TSC booth at your next show.

Kilobaud is the most read magazine in the microcomputer field. It hasn't been difficult to find key industry people reading KB!

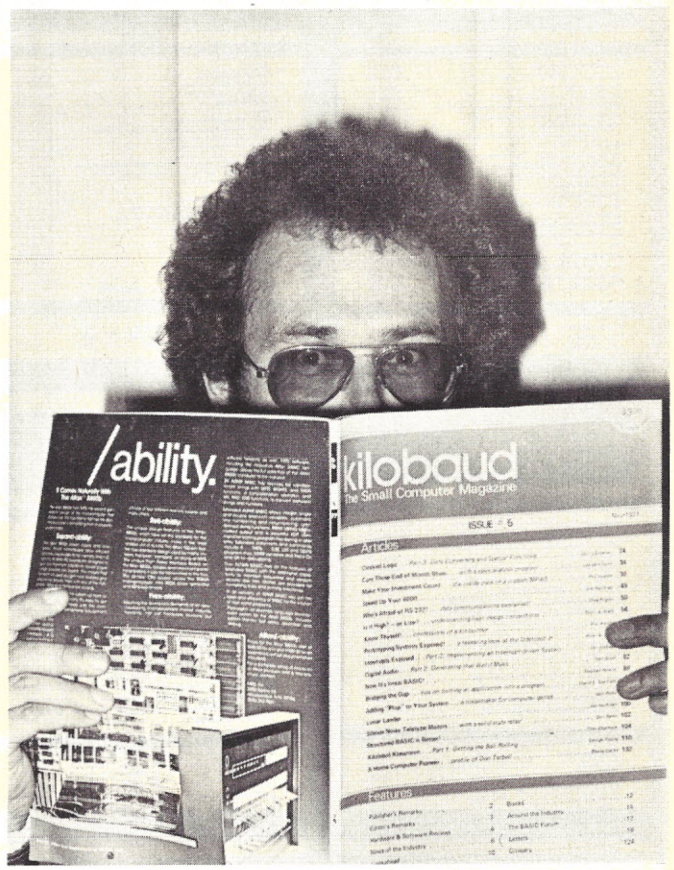
kilobaud

# READ IT YOURSELF!

If you are not yet a subscriber to Kilobaud, we need you and you need us. There is a lot of data in Kilobaud that you will want to have on hand . . . the magazine is like a continuing encyclopedia of microcomputing and programs. You never know when you are desperately going to need something from a past issue . . . and you'll want it immediately. The cost per year is not significant . . . $15 at present for $24 worth of magazines.

You may have noticed that Kilobaud has more articles than any of the other magazines. If you'll keep track of how long it takes you to read Kilobaud as compared to the other computer magazines, you'll see you are getting a great bargain at $15 per year.

# WHO'S BEHIND THE KILOBAUD?




### WHO'S THIS?

Another partly hidden man that you've seen . . . if you've been to any shows. You want a hint, right? Think diskly . . . or waite until the unveiling next month in KILOBAUD.

## SUBSCRIBE TODAY!

**For Instant Subscriptions Call Our Toll Free Number**

1-(800) 258-5473

**Have your credit card handy!**

**Kilobaud ● Peterborough NH 03458**

# it's a good bet the company you bought your computer from doesn't even make peripherals!

It's no great surprise! Most computer companies got their start in the digital logic end of the business. They were great at building calculators and later computers but when it came right down to it, most just didn't have the experience necessary to build the peripherals to support their computer products. And that left a vacuum!

At Heath we had the advantage. Our years of experience in electronic kit design gave us plenty of background with not only digital logic but mechanical and video design as well. And our assembly manuals and documentation are world-famous for easy to understand instructions.

We built the world's first digital color television, a unique fully synthesized FM tuner, digital frequency counters, clocks – even a digital bathroom scale.

So when we entered the personal computing market we had the "know-how" to build not only our outstanding H8 and H11, 8 and 16-bit computers, but, in addition, a complete line of supporting peripheral kits!

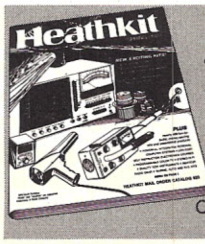
Select the H9 Video Terminal, the H10 Papertape Reader/ Punch, and very soon our own, complete, Floppy Disk system. Each was designed with the systems approach in mind. Each was conceived to integrally mesh with not only our own computers, but through our set of sophisticated interfaces, most others as well. And in that way we're making every effort to fill the vacuum the others left!

So when you're ready to communicate with your computer turn to Heath. We've got the peripheral kits you'll need and at prices you can afford.

Maybe the company who sold you your computer didn't think about peripherals – but we sure did! And come to think about it maybe that's why you should come to Heath... in the first place.

## Heathkit Computers

# FREE Heathkit Catalog




Read about nearly 400 money-saving, fun-to-build electronic kits. Use coupon to send for your mail order catalog or bring coupon to a Heathkit Electronic Center for your catalog.

HEATH Schlumberger — Heath Company, Dept. 351-400, Benton Harbor, MI 49022 — H5

Please send me my FREE Catalog. I am not on your mailing list.

Name__________

Address__________

City__________ State__________

CP-146 Zip__________

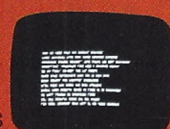

**AVAILABLE LOCALLY AT HEATHKIT ELECTRONIC CENTERS**

(Units of Schlumberger Products Corporation) Retail prices on some products may be slightly higher.

**ARIZONA: Phoenix, 85017,** 2727 W. Indian School Rd., Phone: 602-279-6247; **CALIFORNIA: Anaheim, 92805,** 330 E. Ball Rd., Phone: 714-776-9420; **El Cerrito, 94530,** 6000 Potrero Ave., Phone: 415-236-8870; **Los Angeles, 90007,** 2309 S. Flower St., Phone: 213-749-0261; **Pomona, 91767,** 1555 Orange Grove Ave. N., Phone: 714-623-3543; **Redwood City, 94063,** 2001 Middlefield Rd., Phone: 415-365-8155; **Sacramento, 95825,** 1860 Fulton Ave., Phone: 916-486-1575; **San Diego (La Mesa, 92041),** 8363 Center Dr., Phone: 714-461-0110; **San Jose (Campbell, 95008),** 2350 S. Bascom Ave., Phone: 408-377-8920; **Woodland Hills, 91364,** 22504 Ventura Blvd., Phone: 213-883-0531; **COLORADO: Denver, 80212,** 5940 W. 38th Ave., Phone: 303-422-3408; **CONNECTICUT: Hartford (Avon, 06001),** 395 W. Main St. (Rte. 44), Phone: 203-678-0323; **FLORIDA: Miami (Hialeah, 33012),** 4705 W. 16th Ave., Phone: 305-823-2280; **Tampa, 33614,** 4019 West Hillsborough Ave., Phone: 813-886-2541; **GEORGIA: Atlanta, 30342,** 5285 Roswell Rd., Phone: 404-252-4341; **ILLINOIS: Chicago, 60645,** 3462-66 W. Devon Ave., Phone: 312-583-3920; **Chicago (Downers Grove, 60515),** 224 Ogden Ave., Phone: 312-852-1304; **INDIANA: Indianapolis, 46220,** 2112 E. 62nd St., Phone: 317-257-4321; **KANSAS: Kansas City (Mission, 66202),** 5960 Lamar Ave., Phone: 913-362-4486; **KENTUCKY: Louisville, 40243,** 12401 Shelbyville Rd., Phone: 502-245-7811; **LOUISIANA: New Orleans (Kenner, 70062),** 1900 Veterans Memorial Hwy., Phone: 504-722-6321; **MARYLAND: Baltimore, 21234,** 1713 E. Joppa Rd., Phone: 301-661-4446; **Rockville, 20852,** 5542 Nicholson Lane, Phone: 301-881-5420; **MASSACHUSETTS: Boston (Peabody, 01960),** 242 Andover St., Phone: 617-531-9330; **Boston (Wellesley, 02181),** 165 Worcester Ave. (Rt. 9 just west of Rt. 128), Phone: 617-237-1510; **MICHIGAN: Detroit, 48219,** 18645 W. Eight Mile Rd., Phone: 313-535-6480; **E. Detroit, 48021,** 18149 E. Eight Mile Rd., Phone: 313-772-0416; **MINNESOTA: Minneapolis (Hopkins, 55343),** 101 Shady Oak Rd., Phone: 612-938-6371; **MISSOURI: St. Louis (Bridgeton), 63044,** 3794 McKelvey Rd., Phone: 314-291-1850; **NEBRASKA: Omaha, 68134,** 9207 Maple St., Phone: 402-391-2071; **NEW JERSEY: Fair Lawn, 07410,** 35-07 Broadway (Rte. 4), Phone: 201-791-6935; **Ocean, 07712,** 1013 State Hwy. 35, Phone: 201-775-1231; **NEW YORK: Buffalo (Amherst, 14226),** 3476 Sheridan Dr., Phone: 716-835-3090; **Jericho, Long Island, 11753,** 15 Jericho Turnpike, Phone: 516-334-8181; **Rochester, 14623,** 937 Jefferson Rd., Phone: 716-244-5470; **White Plains (North White Plains, 10603),** 7 Reservoir Rd., Phone: 914-761-7690; **OHIO: Cincinnati (Woodlawn, 45215),** 10133 Springfield Pike, Phone: 513-771-8850; **Cleveland, 44129,** 5444 Pearl Rd., Phone: 216-886-2590; **Columbus, 43229,** 2500 Morse Rd., Phone: 614-475-7200; **Toledo, 43615,** 48 S. Byrne Rd., Phone: 419-537-1887; **PENNSYLVANIA: Philadelphia, 19149,** 6318 Roosevelt Blvd., Phone: 215-288-0180; **Frazer (Chester Co.), 19355,** 630 Lancaster Pike (Rt. 30), Phone: 215-647-5555; **Pittsburgh, 15235,** 3482 Wm. Penn Hwy., Phone: 412-824-3564; **RHODE ISLAND: Providence (Warwick, 02886),** 558 Greenwich Ave., Phone: 401-738-5150; **TEXAS: Dallas, 75201,** 2715 Ross Ave., Phone: 214-826-4053; **Houston, 77027,** 3705 Westheimer, Phone: 713-623-2090; **VIRGINIA: Alexandria, 22303,** 6201 Richmond Hwy., Phone: 703-765-5515; **Norfolk (Virginia Beach, 23455),** 1055 Independence Blvd., Phone: 804-460-0997; **WASHINGTON: Seattle, 98121,** 2221 Third Ave., Phone: 206-682-2172; **WISCONSIN: Milwaukee, 53216,** 5215 W. Fond du Lac, Phone: 414-873-8250.

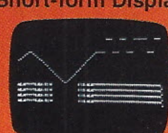
# a computer can get awfully bored when it can't communicate!

**So unless you'd prefer** to have your computer sitting around cooling its chips, we'd strongly suggest you buy a couple of peripherals from Heath.



Long-form Display

Short-form Display



Plot Mode

**Why us? Because we make peripheral kits.** In fact, they're some of the best around. Our H9 is an excellent example. It's a complete ASCII keyboard/12" CRT terminal that was designed for hobbyists just like you. It has a lot of really great features and resolution that's just beautiful. Right on out to 80 characters per line. (Something most outboard TV monitors won't match!) And with built-in selectable interfacing options, the H9 will "converse" with just about any computer going!

**The H10 is another** of our "universal" peripheral kits. Completely self-contained (it even has its own power supply), this rugged paper tape reader/punch gives you quick, convenient mass storage and internal tape duplication capability. It's easy to build and, with its heavy-duty stepper motor, sensitive Darlington photo transistors and precision punches, the H10 is a source of reliable data loading and storage — time after time.

**Best of all,** priced at $530 and $350 respectively,* the H9 and H10 kits cost less than most other comparable peripherals on the market today!

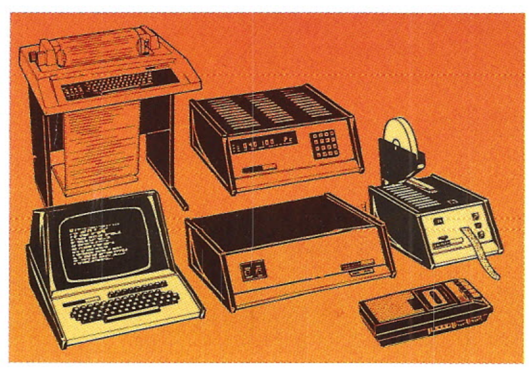
**A computer can get awfully bored** when it can't communicate. Start communicating with yours through an economical peripheral from Heath!

*Mail order, FOB, Benton Harbor, Michigan. Retail prices slightly higher.
Prices and specifications subject to change without notice.

**HEATHKIT COMPUTERS**

**System Engineered for Personal Computing**

# H9 H10



Be sure to use coupon on the facing page of this magazine to order your FREE Heathkit Catalog.


When you put it together, it's really together.
Some people build personal computers for the love of building. The Equinox System™ is for people who build for the love of computing.
You put it together. And it's really together.
The Equinox 100™ mainframe combines the 8080A CPU with a front panel programming station featuring ultra-convenient octal keyboard and digital LED readout.
There are low-cost 4K and 8K memories. All your interfacing comes in one kit. Even EQU/ATE™ and BASIC-EQ™ languages on easy-loading cassettes.
It's all together now. It's all S-100 compatible. And it's upward-compatible with new Equinox™ equipment, software and systems coming in the months ahead.
See the Equinox System™ at your local computer shop. Call toll-free to 800-648-5311 (BAC/MC accepted). Or write Equinox Division, Parasitic Engineering, P.O. Box 6314, Albany, California 94706. THE EQUINOX SYSTEM™ All together now.
EQUINOX 100